13. Instruction Timings

Section 12, The TMS34010 Instruction Set, describes each GSP instruction, including instruction cycle timings. This section provides details pertaining to instruction timings for the following groups of instructions:

Section **Page**
13.1 General Instructions 13-2
13.2 MOVE and MOVB Instructions 13-4
13.3 FILL Instructions 13-9
13.4 PIXBLT Instructions 13-16
13.5 PIXBLT Expand Instructions 13-26
13.6 The LINE Instruction 13-34

13.1 General Instructions

General instructions include all GSP instructions *except* MOVEs, MOVBs, FILLs, PIXBLTs, and LINE.

Each instruction description in Section 12 contains a **Machine States** field that lists the number of CPU states required to execute the instruction. This description appears as:

Machine States <cache hit case>, <cache disabled case>

These two values represent the number of CPU states required to execute the instruction for each of two cases:

- The **cache hit case** gives the number of execution states if the instruction and its extension words reside entirely in cache. Thus, only actual execution states (using the CPU) and external memory cycles for data transfer are counted with the instruction.

- The **cache disabled case** gives the number of execution states if the cache is disabled when the instruction is executed. In this case, external memory cycles for fetching the instruction word and any extension words are counted with the instruction in addition to states through the CPU and memory states for data transfer. Cache is usually only disabled during debugging.

Cache disabled timing is not necessarily worst case timing. It may sometimes be exceeded when the cache is enabled but the instruction is not in the cache (this is known as a *cache miss*).

13.1.1 Best Case Timing - Considering Hidden States

Best case timing occurs when an instruction is executed entirely in parallel with the end of a previous instruction. According to some microprocessor conventions, many TMS34010 instructions would have a best case timing of 0 states. Since this is unrealistic, the convention used here assigns a finite (nonzero) timing value but allows for instruction overlap by using the concept of *hidden states.*

Hidden states are memory write cycles that occur at the end of a given instruction. Parallelism is achieved when the CPU is executing instructions at the same time the memory controller is writing to memory. The machine states consumed by the instructions that the CPU is executing hide the machine states consumed by the write cycles. These hidden machine states are not counted against the instruction that incurs them, but are counted against subsequent instructions. If an instruction uses the local bus before all of the hidden cycles have been overlapped by subsequent instructions, that instruction must wait for the hidden cycles to complete. Up to nine machine states may be hidden by write cycles incurred by a single instruction.

In the timing charts in this section and in the **Machine States** portions of the instruction descriptions, hidden states are indicated by parentheses as shown below:

Machine States <cache hit case>+(<hidden states>), <cache disabled case>

13.1.2 Other Effects on Instruction Timing

Instruction timing varies, depending on:

- Whether the cache is enabled
- Whether the instruction and extension words are in cache or not
- The field size and the word alignment of memory data manipulated by the instruction

The timing for some instructions (particularly the MOVE, MOVEB, LINE, FILL, and PIXBLT instructions) is affected by the values of implied operands and on the alignment and field sizes of any associated memory accesses.

In addition, several system-dependent factors that are not included in timing values may further influence the instruction timings:

- Wait states on the local memory bus
- Host accesses via the host port
- Display refresh operations
- DRAM refresh operations
- HOLD/HLDA accesses

13.2 MOVE and MOVB Instructions

Timings for MOVE and MOVB instructions are in the following tables:

Table **Page**

13-1 MOVE and MOVB Memory-to-Register Timings 13-5
13-2 MOVE and MOVB Register-to-Memory Timings 13-6
13-4 MOVE Memory-to-Memory Timings 13-7

MOVE and MOVB instructions are field operations, so their timings are affected by factors such as memory address, field size, and field extensions. These factors define the field alignment, which in turn defines the number of memory states required to insert or extract the field from memory. Figure 13-1 illustrates seven cases of alignment, labelled A–G, that are used in the MOVE and MOVB timing tables.

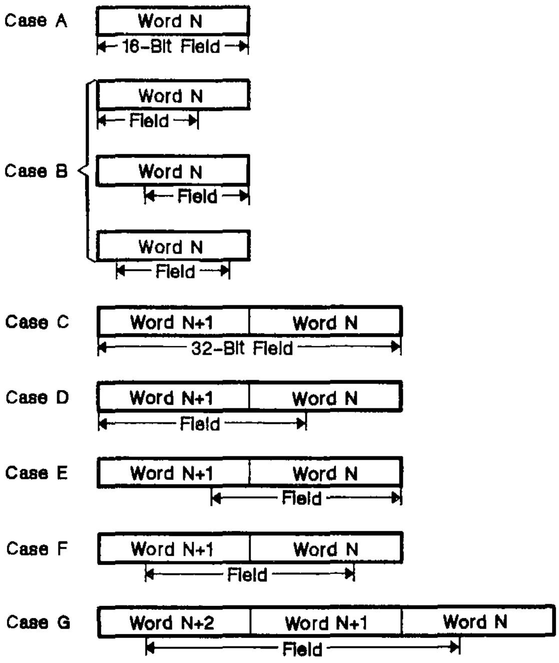

Figure 13-1. Field Alignments in Memory

Case A A 16-bit field is aligned on word boundaries.

Cases B1–B3
The field length is less than 16 bits.

- In **Case B1**, the field starting address is not aligned to a word boundary, although the end of the field coincides with the end of the word.
- In **Case B2**, the field starting address is aligned to a word boundary, but the end of the field does not coincide with the end of the word.
- In **Case B3**, the field length is 14 bits or less, and neither the start nor the end of the field is aligned to a word boundary.

Case C A 32-bit field is aligned on word boundaries.

Case D The field size is greater than 16 bits. The field starting address is not aligned to a word boundary, although the end of the field coincides with the end of a word.

Case E The field size is greater than 16 bits. The field starting address is aligned to a word boundary, but the end of the field does not coincide with the end of a word.

Case F The field straddles the boundary between two words. Neither the start nor the end of the field is aligned to a word boundary.

Case G The field size ranges from 18 to 32 bits, and the field straddles two word boundaries. Neither the start nor the end of the field is aligned to a word boundary.

13.2.1 Moves Between Registers and Memory

Table 13-1 lists the timing for memory-to-register moves for each case of the destination alignment in Figure 13-1. Table 13-2 lists the timing for register-to-memory moves. Note that there are no hidden states for memory-to-register moves.

Table 13-1. MOVE and MOVB Memory-to-Register Timings

Instruction	Field Alignment Type		
	A or B	C, D, E, F	G
MOVB *Rs,Rd	3,6	5,8	N/A
MOVB *Rs(Disp),Rd	5,11	7,13	N/A
MOVB @Address,Rd	5,14	7,16	N/A
MOVE *Rs,Rd	3,6	5,8	7,10
MOVE *Rs+,Rd	3,6	5,8	7,10
MOVE -*Rs,Rd	4,7	6,9	8,11
MOVE *Rs(Disp),Rd	5,11	7,13	9,15
MOVE @Address,Rd	5,14	7,16	9,19

Notes: 1. Add 1 state to MOVES for sign extension.
2. The first number specifies the number of cycles required when the entire instruction is contained within cache (cache hit case). The second number specifies the number of cycles required when the cache is disabled (cache disabled case).

Table 13-2. MOVE and MOVB Register-to-Memory Timings

Instruction	Field Alignment Type				
	A	B or C	D or E	F	G
MOVB Rs,*Rd	N/A	1+(3),7	N/A	1+(7),11	N/A
MOVB Rs,*Rd(Disp)	N/A	3+(3),7	N/A	3+(7),13	N/A
MOVB Rs,@Address	N/A	1+(3),7	N/A	3+(7),13	N/A
MOVE Rs,*Rd	1+(1),5	1+(3),7	1+(5),9	1+(7),11	1+(9),13
MOVE Rs,*Rd+	1+(1),5	1+(3),7	1+(5),9	1+(7),11	1+(9),13
MOVE Rs,-*Rd	2+(1),6	2+(3),8	2+(5),10	2+(7),12	2+(9),14
MOVE Rs,*Rd(Disp)	3+(1),7	3+(3),9	3+(5),11	3+(7),13	3+(9),15
MOVE Rs,@Address	3+(1),7	3+(3),9	3+(5),11	3+(7),13	3+(9),15

Note: The first number specifies the number of cycles required when the entire instruction is contained within cache (cache hit case). The second number specifies the number of cycles required when the cache is disabled (cache disabled case). Hidden states are indicated by parentheses.

13.2.2 Memory-to-Memory Moves

Table 13-4 lists memory-to-memory move timings for each combination of source and destination alignment. Table 13-3 lists numeric indices which are used in Table 13-4. The indices are associated with each source and destination alignment pair (the alignments are shown in Figure 13-1 on page 13-4). To use these tables:

1) Determine the source and destination alignment,
2) Locate the alignment and its index in Table 13-3, and
3) Use the index to select the correct column for a particular MOVE addressing mode in Table 13-4.

Table 13-3. Alignment Indices for Memory-to-Memory Moves

Source Field Alignment	Destination Field Alignment						
	A	B	C	D	E	F	G
A	1	–	–	–	–	3	–
B	–	2	–	–	–	3	–
C	–	–	6	–	–	–	9
D	–	–	–	7	7	8	9
E	–	–	–	7	7	8	9
F	4	5	–	7	7	8	9
G	–	–	10	11	11	12	13

Table 13-4. MOVE Memory-to-Memory Timings

Instruction	Memory-to-Memory Index – Source to Destination						
	1	2	3	4	5	6	7
MOVB *Rs,*Rd	N/A	3+(3),7	3+(7),13	N/A	5+(3),11	N/A	N/A
MOVB *Rs(D),*Rd(D)	N/A	5+(3),7	5+(7),21	N/A	6+(3),13	N/A	N/A
MOVB @SAddr,@DAddr	N/A	7+(3),7	7+(7),29	N/A	6+(3),12	N/A	N/A
MOVE *Rs,*Rd	3+(1),7	3+(3),9	3+(7),13	5+(1),9	5+(3),11	5+(3),11	5+(5),13
MOVE *Rs+,*Rd+	4,7	4+(2),9	4+(6),13	6,9	6+(2),11	6+(2),11	6+(4),13
MOVE -*Rs,-*Rd	4+(1),8	4+(3),10	4+(7),14	6+(1),10	6+(3),12	6+(3),12	6+(5),14
MOVE *Rs(S),*Rd+	5+(1),12	5+(3),14	5+(7),18	7+(1),14	7+(3),16	7+(3),13	7+(5),15
MOVE *Rs(S),*Rd(D)	5+(1),15	5+(3),17	5+(7),21	7+(1),17	7+(3),19	7+(3),16	7+(5),18
MOVE @SAddr,*Rd+	5+(1),15	5+(3),17	5+(7),21	7+(1),17	7+(3),19	7+(3),16	7+(5),18
MOVE @SAddr,@DAddr	7+(1),23	7+(3),25	7+(7),29	9+(1),25	9+(3),27	9+(3),24	9+(5),26

Instruction	Memory-to-Memory Index – Source to Destination					
	8	9	10	11	12	13
MOVB *Rs,*Rd	5+(7),15	N/A	N/A	N/A	N/A	N/A
MOVB *Rs(D),*Rd(D)	7+(7),19	N/A	N/A	N/A	N/A	N/A
MOVB @SAddr,@DAddr	9+(7),27	N/A	N/A	N/A	N/A	N/A
MOVE *Rs,*Rd	5+(7),15	5+(9),17	7+(3),13	7+(5),15	5+(7),17	9+(9),21
MOVE *Rs+,*Rd+	6+(6),15	6+(8),17	8+(2),13	8+(4),15	6+(6),17	10+(8),21
MOVE -*Rs,-*Rd	6+(7),15	6+(9),18	8+(3),14	8+(5),16	6+(7),18	10+(9),22
MOVE *Rs(S),*Rd+	7+(7),16	7+(9),19	9+(3),18	9+(5),20	7+(7),22	11+(9),26
MOVE *Rs(S),*Rd(D)	7+(7),19	7+(9),22	9+(3),21	9+(5),23	7+(7),25	11+(9),29
MOVE @SAddr,*Rd+	7+(7),19	7+(9),22	9+(3),21	9+(5),23	7+(7),25	11+(9),29
MOVE @SAddr,@DAddr	9+(7),27	9+(9),30	11+(3),29	11+(5),31	9+(7),33	13+(9),37

Note: The number on the left specifies the number of cycles required when the entire instruction is contained within cache (cache hit case). The number on the right specifies the number of cycles required when the cache is disabled (cache disabled case). Hidden states are indicated by parentheses.

13.2.3 MOVE Timing Example

Figure 13-2 illustrates a `MOVE @SAddress,@DAddress,0` instruction with these initial implied operands:

DADDR	=	>161
SADDR	=	>E5
FS0	=	>31
FE0	=	don't care

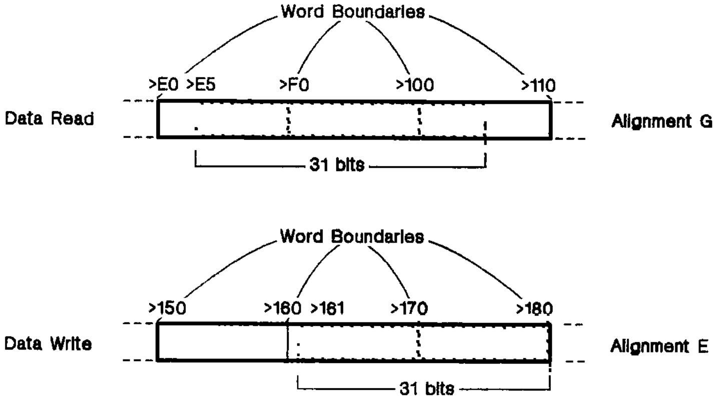

Figure 13-2. MOVE Timing Example

As Figure 13-2 shows, this is a memory-to-memory move with a field size of 31 bits. The source data begins at address >E5 and spans three words; as Figure 13-1 (page 13-4) shows, the source data has alignment G. The destination location begins at address >161 and spans two words; as Figure 13-1 shows, the destination alignment is type E. Table 13-3 (page 13-6) shows the source-to-destination alignment indices for Table 13-4; alignment G to E points to column 11 in Table 13-4. By locating the entry for a `MOVE @SAddress,@DAddress` in column 11 of Table 13-4, we see that the timing for this example is **11+(5),31**.

Thus, this MOVE example would consume 11 machine states (plus 5 hidden states) if this code resided in cache. If the instruction cache was not enabled, this example would consume 31 machine states. The memory accesses at the end of the MOVE consume 5 machine states, which may be hidden by subsequent cache-resident instructions.

13.3 FILL Instructions

The total time for the FILL instruction is calculated by adding a setup time to a transfer time:

FILL time = FILL setup time + FILL transfer time

- The **setup sequence** executes an initialization sequence, performing any necessary setup operations and translations. (This may include XY to linear conversions and window preclipping.) The result of the setup includes the dimensions of the array that is to be moved.

- The **transfer sequence** performs the actual data transfer from the source register to the destination array.

FILL setup and transfer timings are in the following tables:

Table **Page**
13-5 FILL Setup Time 13-9
13-6 FILL Transfer Timing† 13-10

13.3.1 FILL Setup Time

FILL setup time is the overhead incurred by the FILL instructions from performing initialization, XY conversions, and window operations. Window operations are performed before the FILL transfer begins. Window options that affect FILL setup timing include:

- No window clipping (*W=0*)
- A window clip that requires no change (*array fits*)
- A window clip that affects the starting pointer (*start adjust*)
- A window clip that affects the array transfer dimensions (*dimension adjust*)
- A window clip that affects both the starting and the ending pointers (*adjust both*)
- A window *miss* requesting an interrupt
- A window *hit*

Table 13-5 illustrates the effects of windowing operations on FILL setup timing. Corner adjust operations have no effect on FILL setup timing.

Table 13-5. FILL Setup Time

	Window Operation							Corner Adjust		
Instruction	W=0	Array Fits	Start Adjust	Dimens Adjust	Adjust Both	Miss	Hit	PBH=1 PBV=0	PBH=0 PBV=1	PBH=1 PBV=1
FILL L	4	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
FILL XY	6	9	16	12	20	N/A	N/A	N/A	N/A	N/A

Note: These timings are for the cache hit case; add 3 machine states for cache disabled timing.

For example, a FILL XY with preclipping that requires both the starting and ending array corners to be adjusted would consume 20 states of setup time.

13.3.2 FILL Transfer Timing

Table 13-6 lists FILL transfer timings. Transfer timing is the time required (in addition to the setup time) to execute the actual data transfer to memory. Transfer timing is based on several parameters such as the number of rows in the adjusted array (*L*), the number of words affected per row (*N*), graphics operations (*G*), and four possible destination array alignments (A, B, C, and D). These factors are described in the list that follows the table.

Table 13-6. FILL Transfer Timing†

Line Length	Array Alignments			
	A	B	C	D
Short (N=1)	$(1+G)L+2$	$(2+G)L+2$	$(2+G)L+1$	$(2+G)L+1$
Medium (N=2)	$(2+2G)L+2$	$(3+2G)L+2$	$(3+2G)L+2$	$(4+2G)L+1$
Long ($N\geq3$)	$(1+NG)L+2$	$(2+NG)L+5$	$(3+NG)L+2$	$(4+NG)L+1$

† Subtract any alignment/graphics adjustment from these values

Key:

L Number of rows (see page 13-10)
N Number of words per row (see page 13-12)
G Value derived from selected graphics operation (see Table 13-7 on page 13-12)

- **Number of Rows in the Adjusted Array (*L*)**

  The working dimensions (*L* rows × *M* pixels) for the fill are determined by the originally supplied destination pointer (DADDR) and dimensions (DYDX) in conjunction with window preclipping.

- **Alignment of Leading and Trailing Words in Rows**

  After clipping, the data transfer portion of the FILL treats the array as a series of *L* rows of *M* pixels. These *M* pixels are spread across *N* words in each row of the destination array. Figure 13-3 illustrates a single row of a destination array in memory. The FILL algorithm resolves rows into three portions:

  1) The leading edge at the beginning of the row
  2) The center *N*-2 words of the row
  3) The trailing edge at the end of the row

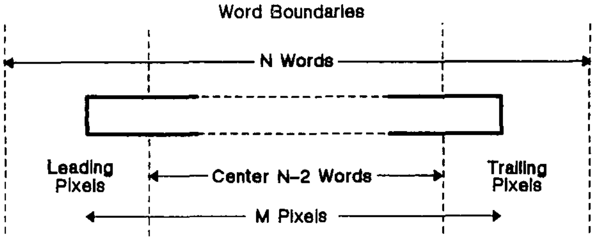

Figure 13-3. Pixel Block Alignment in X

As Figure 13-3 shows, a row of *N* words includes one word each for the leading and trailing parts of the transfer and *N*-2 words for the center portion. The FILL always transfers the center portion of the row as a series of 16-bit words. Thus, the alignment of the leading and trailing words in the row characterize the alignment type of the array. Figure 13-4 illustrates the four possible alignments (A, B, C, and D) of destination array rows within pixel blocks in memory.

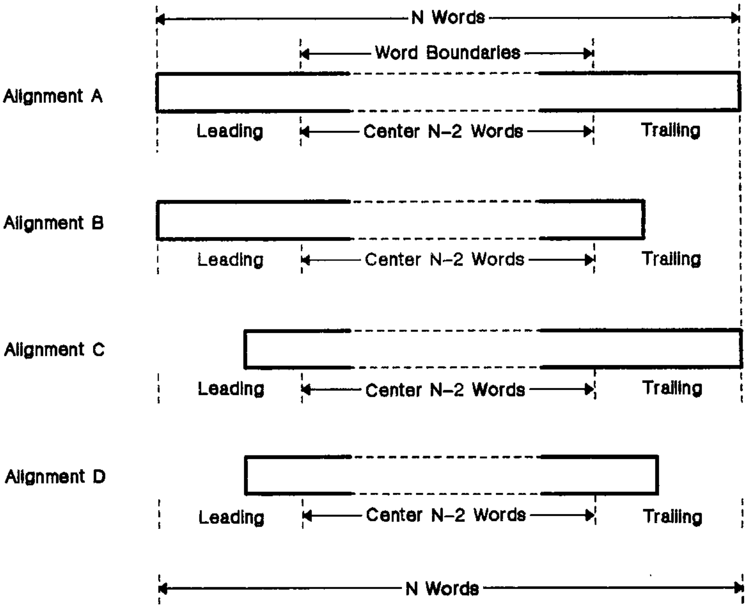

Figure 13-4. Pixel Block Alignments

Word alignment is constant from row to row because DPTCH is constrained to be a multiple of 16 for most FILLs. If a FILL is only one pixel wide, and all the rows are contained in single words in memory, DPTCH may be any value. If DPTCH is not a multiple of 16, word alignment may vary between cases B, C, and D. Average timing for this situation may be derived using alignment C. Worst case timing for this situation may be derived using alignment D.

- **Row Length (Number of Words *N* per Row)**

  Row length is determined by a combination of the computed array pointer value in DADDR, the clipped DX dimension, and the pixel size stored in the PSIZE register. The data transfer algorithm breaks down into one of three cases, short, medium, or long, according to the number of words *N* in a row. These three cases include:

  > **Short case**. The destination array row occupies only one word in memory (N=1). In this case, only one write (or read-modify-write) operation is required to place the row into the destination array. Alignment for the short case is either type A for exactly aligned arrays or type B, C, or D for nonaligned arrays (which require a read-modify-write).
  >
  > **Medium case**. The destination row occupies two words in memory (N=2). In this case, the row has no center portion and the array alignment is determined by the alignments of the first and last words in the row.
  >
  > **Long case**. The destination row occupies all or part of at least three words ($N \geq 3$). This is the general case for array alignment discussions.

- **Transfer Direction in X**

  Transfer direction does not apply to FILLs. FILL transfers proceed a single word of pixels at a time in the order of increasing X and increasing Y. This corresponds to a transfer from left-to-right and top-to-bottom for the default screen orientation.

- **Selected Graphics Operations (*G*)**

  Graphics operations such as plane masking, transparency, and pixel processing influence FILL transfer timing because the destination pixels must be read before they are replaced. However, the effects of these operations vary because they are performed by different portions of the TMS34010 hardware. For instance, plane masking, transparency, and field insertion are all performed by the GSP memory controller; any combination of these operations uses 2 machine states for each word written. Pixel processing, on the other hand, is performed by the GSP CPU, and requires 2, 4, 5, or 6 states per word (independently of other operations). *The minimum cycle time for any graphics operation,* then, is **2 machine states** (one memory cycle) using the pixel processing *replace* operation, with plane masking and transparency disabled. Table 13-7 shows these values.

Table 13-7. Timing Values per Word for Graphics Operations (G)

	Pixel Processing Operation			
Graphics Operation	**Replace**	**Other Booleans or ADD**	**ADDS,SUB MAX or MIN**	**SUBS**
No plane masking or transparency	2	4	5	6
Read-modify-write, plane masking, or transparency	4	6	7	8

- **Alignment/Graphics Adjustment**

  An additional adjustment may be necessary when plane masking or transparency are enabled and the alignment type is B, C, or D. As the second line of Table 13-7 shows, if a particular word in a destination row has already been read as part of a read-modify-write operation, no **additional** states are required to perform *plane masking* or *transparency* for that word. Since the alignment types with misaligned edges (B, C, and D) already assume a RMW (read-modify-write) on their respective edges, the effect of plane masking or transparency can be ignored for these edges. That is, after you have calculated the timing using the proper value for the graphics operation, you can **subtract** 2 states (cases B and C) or 4 states (case D) per row from the transfer timings for the respective alignment cases. Case A requires no adjustment.

13.3.3 FILL Timing Examples

FILL timing is calculated by adding the FILL setup value to the FILL transfer value:

FILL time = FILL setup time + FILL transfer time - alignment adjustment

FILL setup timings, transfer timings, and the effects of graphics operations are in the following tables:

Table		Page
13-5	FILL Setup Time	13-9
13-6	FILL Transfer Timing†	13-10
13-7	Timing Values per Word for Graphics Operations (G)	13-12

The following three examples illustrate timing for a **FILL XY** with these initial implied operands:

DADDR	=	>004400E4	(X=228, Y=68)
DPTCH	=	>800	(X extent = 512 pixels x 4 bits per pixel)
OFFSET	=	>0	
WSTART	=	>004900EB	(X=235, Y=73)
WEND	=	>005F0140	(X=320, Y=95)
DYDX	=	>0014003C	(DX=60, DY=20)
PSIZE	=	>4	
CONVDP	=	>14	(LMO DPTCH)

The setup and transfer timings for these examples are the same, except each uses a different graphics operation. Figure 13-5 illustrates the destination array and window used in these examples. The shaded portion is the area of intersection.

- **Setup Time**: W=3 is the window preclipping option. This option requires the starting corner to be adjusted. As Table 13-5 shows, the setup time for a FILL XY with these options is 16 machine states.
- **Transfer Time:** As Figure 13-5 shows, the window preclipping results in an array with a Y dimension of 15 (L=15). The resulting X dimension is 53 pixels and the pixel size is 4; 53 divided by 4 produces 13.25, so N=14. Since this N is greater than 3, this example conforms to the long case. The trailing edge is word aligned but the leading edge is not, so the alignment type is C. As Table 13-6 shows, the transfer time for a FILL XY with these characteristics is $(3+NG)L + 2$.

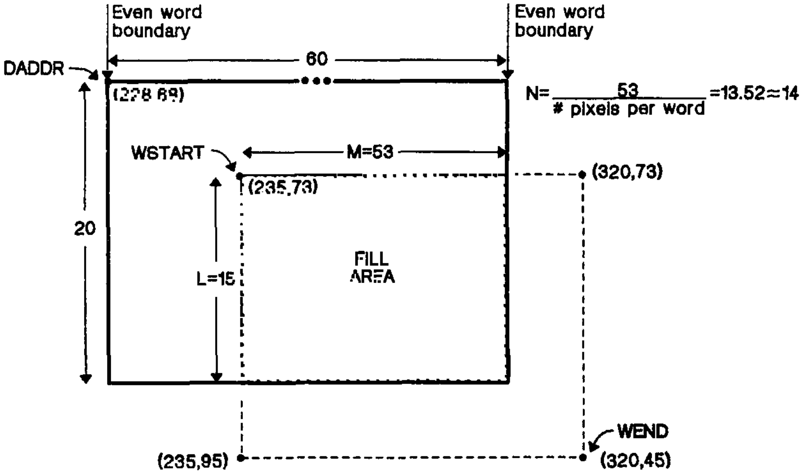

Figure 13-5. FILL XY Timing Example

Example 13-1. W=3, T=0, PP=0, No Plane Masking

The pixel processing *replace* operation has been selected, and transparency and plane masking are not enabled. According to Table 13-7, ***G*=2**. The FILL timing for this instruction can be calculated as follows:

$$\begin{aligned} \text{FILL time} &= \text{FILL setup time} &&+ \text{FILL transfer time} \\ &= \text{Adjust pointer} &&+ (3+NG)L + 2 \\ &= 16 &&+ [(3 + (14\times 2)]15 + 2 \\ &= 483 \text{ states} \end{aligned}$$

The FILL writes 795 pixels (at four bits per pixel) in these 483 states.

Example 13-2. W=3, T=0, PP=20, No Plane Masking

The pixel processing MAX operation has been selected, and transparency and plane masking are not enabled. According to Table 13-7, ***G*=5**. The FILL timing is now calculated as:

$$\begin{aligned}
\text{FILL time} &= \text{FILL setup time} &&+ \text{FILL transfer time}\\
&= \text{Adjust pointer} &&+ (3+NG)L + 2\\
&= 16 &&+ [3 + (5\times14)]15 + 2\\
&= 1{,}113 \text{ states}
\end{aligned}$$

This FILL requires 1,113 states to merge the pixel values in the COLOR1 register with those in the destination array using the MAX operation. The portion of the array lying within the window contains 795 pixels.

Example 13-3. W=3, PP=5, T=1, Plane Masking Enabled

The pixel processing XNOR operation has been selected, and transparency and plane masking **are** enabled. According to Table 13-7, ***G*=6**. Alignment type C incurs a read-modify-write at the leading edge of each row. The extra read included in the RMW can be used by the plane masking or transparency hardware, so an alignment/graphics adjustment is necessary. The adjustment negates the effect of the extra read cycles in each row that are attributed to the graphics operations. For this example, the amount subtracted is 2 (the number of machine states for a read cycle) times *L* (the number of rows). The FILL timing is now calculated as:

$$\begin{aligned}
\text{FILL time} &= \text{FILL setup time} &&+ \text{FILL transfer time} &&- \text{adjustment}\\
&= \text{Adjust pointer} &&+ (3+NG)L + 2 &&- 2L\\
&= 16 &&+ [3 + (6\times14)]15 + 2 &&- (2\times15)\\
&= 1{,}293 \text{ states}
\end{aligned}$$

This FILL requires 1,293 states to write the XNOR of the pixel values in the COLOR1 register with those in the destination array for 795 pixels (at four bits per pixel).

13.3.4 Interrupt Effects on FILL Timing

The FILL instruction may be interrupted on a word boundary during the transfer portion of the FILL algorithm. It can also be interrupted at the end of each row. The context of the FILL is saved in reserved registers, and the PBX bit is set in the copy of the status register that is pushed onto the stack. The worst case latency caused by an interrupt is 20 machine states for the interrupt to be recognized. The time for the context switch must be added to this. See Section 8.4.1, Interrupt Latency (page 8-5) for context switch information.

13.4 PIXBLT Instructions

PIXBLT instructions covered in this section include:

- PIXBLT L,L
- PIXBLT XY,L
- PIXBLT L,XY
- PIXBLT XY,XY

(PIXBLT B,L and PIXBLT B,XY are covered in Section 13.5.)

The total PIXBLT instruction timing is obtained by adding a setup time to a transfer time:

PIXBLT time = PIXBLT setup time + PIXBLT transfer time

- The **setup sequence** executes an initialization sequence, performing any necessary setup operations and translations. (This includes XY-to-linear conversion and window preclipping.) The result of the setup includes the dimensions of the source array.
- The **transfer sequence** performs the actual data transfer from the source array to the destination array.

PIXBLT setup and transfer timings are in the following tables:

Table		Page
13-8	PIXBLT Setup Time	13-16
13-9	PIXBLT Transfer Timing†	13-18

13.4.1 PIXBLT Setup Time

Table 13-8 lists PIXBLT setup times. Setup time is the overhead incurred by the PIXBLT instructions in performing initialization, XY conversions, window options, and corner adjust. Setup time is affected by both the window and corner adjust operations. The effects of these operations are described in the list that follows Table 13-8.

Table 13-8. PIXBLT Setup Time

	Window Operation							Corner Adjust		
Instruction	W=0	Array Fits	Start Adjust	Dimens Adjust	Adjust Both	Miss	Hit	PBH=1 PBV=0	PBH=0 PBV=1	PBH=1 PBV=1
PIXBLT L,L	7	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
PIXBLT XY,L	9	N/A	N/A	N/A	N/A	N/A	N/A	+1	+2	+4
PIXBLT L,XY	9	12	19	15	23	N/A	N/A	+1	+2	+4
PIXBLT XY,XY	12	15	22	18	26	N/A	N/A	+1	+2	+4

For example, consider a PIXBLT XY,XY instruction with preclipping that requires both the starting and ending array corners to be adjusted (PBH=1 and PBV=0). The setup timing for this example would be 26+1=27 states.

- **Window Operations**

  Window operations are performed before the PIXBLT transfer begins. Window options that affect PIXBLT setup timing include:

  - No window checking (*W=0*)
  - A window clip that requires no change (*array fits*)
  - A window clip that affects the starting pointer (*start adjust*)
  - A window clip that affects the array transfer dimensions (*dimension adjust*)
  - A window clip that affects both the starting and ending pointers (*adjust both*)
  - A window *miss* that requests an interrupt
  - A window *hit*

- **Corner Adjust (PBH and PBV)**

  The TMS34010 may need to adjust the starting corner of the source and destination arrays for the PIXBLT L,XY, PIXBLT XY,L, and PIXBLT XY,XY instructions. The default starting corner is the upper left corner of the array. This can be altered by changing the values of the PBH and PBV (PIXBLT horizontal and vertical) bits. Possible corner adjustments (with default origin ORG=0) include:

  - No corner adjust (PBH=0, PBV=0)
  - Adjust to upper right corner (PBH=1, PBV=0)
  - Adjust to lower left corner (PBH=0, PBV=1)
  - Adjust to lower right corner (PBH=1, PBV=1)

  The TMS34010 adjusts corners before PIXBLT execution begins. For each combination of PBH and PBV, the GSP adjusts the source and destination starting address pointers to point to the appropriate corner of the arrays. This assures that the same pixel block is moved, despite the difference in X and Y transfer directions.

  The original source and destination pointers must be supplied through software. The pointers should indicate the least significant pixel in the array, except for PIXBLT L,L. For this instruction, the PBH and PBV bits affect only the *direction* of the move; the GSP does not adjust the starting corner.

13.4.2 PIXBLT Transfer Timing

Table 13-9 lists PIXBLT transfer timings. Transfer timing is the time required (in addition to the setup time) to execute the actual data transfer to memory. Transfer timing is affected by several factors, including the number of rows in the adjusted array (L), the number of words affected per row (N), graphics operations (G), and four possible destination array alignments (A, B, C, and D). These factors are described in the list that follows the table.

Table 13-9. PIXBLT Transfer Timing†

PBH = 0				
Row Lengths and Alignment	**Destination Array Alignment**			
	A	**B**	**C**	**D**
Short (N=1)				
$D \geq S$	$(G+4)L + 5$	$(G+6)L + 3$	$(G+6)L + 3$	$(G+6)L + 3$
$D < S$	$(G+4)L + 5$	$(G+6)L + 3$	$(G+6)L + 3$	$(G+6)L + 3$
Medium (N=2)				
$D \geq S$	$[2+(4+2G)]L + 5$	$[4+(4+2\ G)]L + 3$	$[4+(4+2\ G)]L + 5$	$[6+(4+2\ G)]L + 3$
$D < S$	$[4+(4+2G)]L + 4$	$[6+(4+2\ G)]L + 2$	$[6+(4+2\ G)]L + 4$	$[8+(4+2\ G)]L + 2$
Long ($N \geq 3$)				
$D \geq S$	$[(2+\ G)N]L + 5$	$[2+(2+\ G)N]L + 3$	$[2+(2+\ G)N]L + 5$	$[2+(4+\ G)N]L + 3$
$D < S$	$[2+(2\ G)N]L + 4$	$[4+(2\ G)N]L + 2$	$[4+(2\ G)N]L + 4$	$[6+(2\ G)N]L + 2$
PBH = 1				
Row Lengths and Alignment	**Destination Array Alignment**			
	A	**B**	**C**	**D**
Short (N=1)				
$D \geq S$	$(G+3)L + 8$	$(G+4)L + 7$	$(G+4)L + 7$	$(G+4)L + 7$
$D < S$	$(G+3)L + 8$	$(G+4)L + 7$	$(G+4)L + 7$	$(G+4)L + 7$
Medium (N=2)				
$D \geq S$	$[2+(4+2\ G)]L + 4$	$[4+(4+2\ G)]L + 3$	$[4+(4+2\ G)]L + 4$	$[6+(4+2\ G)]L + 3$
$D < S$	$[4+(4+2\ G)]L + 5$	$[5+(4+2\ G)]L + 4$	$[6+(4+2\ G)]L + 5$	$[7+(4+2\ G)]L + 4$
Long ($N \geq 3$)				
$D \geq S$	$[1+(2+\ G)N]L + 4$	$[3+(2+\ G)N]L + 3$	$[3+(2+\ G)N]L + 4$	$[5+(2+\ G)N]L + 3$
$D < S$	$[3+(2+\ G)N]L + 5$	$[4+(2+\ G)N]L + 4$	$[5+(2+\ G)N]L + 5$	$[6+(2+\ G)N]L + 4$

† Subtract any alignment/graphics adjustment from these values

Key:

L	Number of rows in the array (see page 13-19)
N	Number of destination words per row (see page 13-20)
G	Value dependent on selected graphics operation (see Table 13-10 on page 13-22)
$D \geq S$	First destination to source alignment case (see page 13-21)
$D < S$	Second destination to source alignment case (see page 13-21)

- **Number of Rows in the Array (*L*)**

  The working dimensions (*L* rows by *N* words) for the block transfer are determined by the original destination pointer (DADDR) and dimensions (DYDX) in conjunction with window preclipping. *L* represents the number of rows in the clipped array.

- **Alignment of Leading and Trailing Words in Rows**

  After clipping, the data transfer portion of the PIXBLT treats the array as a series of *L* rows of *M* pixels. These *M* pixels are spread across *N* words in each row of the destination array. *N* and *L* affect the transfer timing. Alignment does not vary from row to row because DPTCH is constrained to be a power of two.

  Figure 13-6 illustrates a single row of a destination array in memory. The PIXBLT algorithm resolves rows into three portions:

  1) The leading edge at the beginning of a row

  2) The center *N*-2 words of the row

  3) The trailing edge at the end of the row

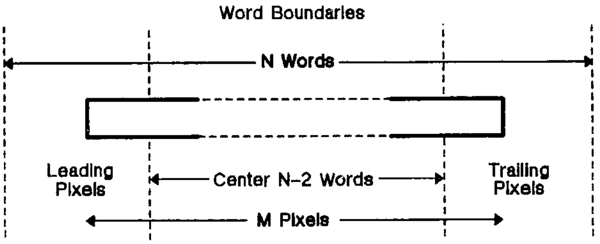

Figure 13-6. Pixel Block Alignment in X

As Figure 13-6 shows, a row of *N* words includes one word each for the leading and trailing parts of the transfer and *N*-2 words for the center portion. The PIXBLT always transfers the center portion of the row as a series of 16-bit words. Thus, the alignment of the leading and trailing portions characterize the alignment type of the array. Figure 13-7 illustrates the four possible alignments (A, B, C, and D) of a destination array.

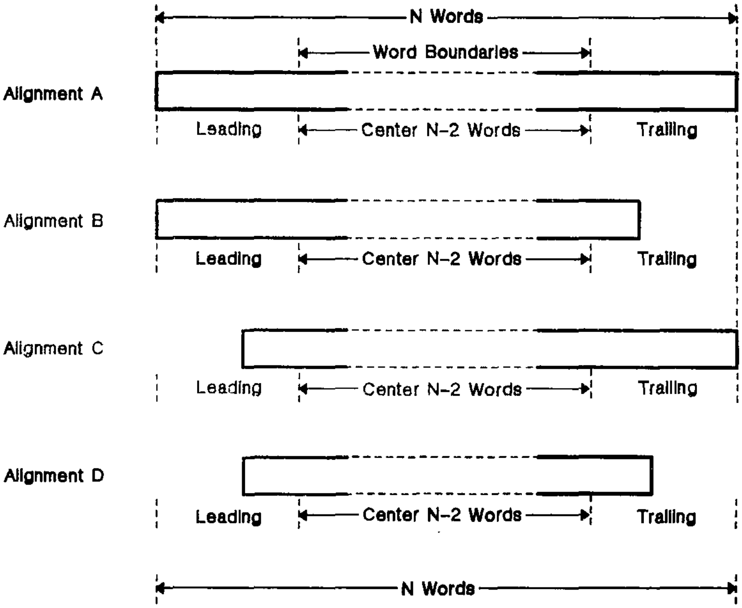

Figure 13-7. Pixel Block Alignments

- **Row Length (Number of Words *N* per Row)**

  Row length is determined by a combination of the computed array pointer value in DADDR, the clipped DX dimension, and the pixel size stored in the PSIZE register. The data transfer algorithm breaks down into one of three cases, short, medium, or long, according to the number of words *N* in a row. These three cases include:

  Short case. The destination array row occupies only one word in memory (*N*=1). In this case, only one write (or read-modify-write) operation is required to place the row into the destination array. Alignment for the short case is either type A for exactly aligned arrays or type B, C, or D for nonaligned arrays (which require a read-modify-write).

  Medium case. The destination row occupies two words in memory (*N*=2). In this case, there is no center portion to the row and the array alignment is determined by the alignments of the first and last words in the row.

  Long case. The destination row occupies all or part of at least three words (*N*≥3). This is the general case for array alignment discussions.

- **Relative Alignment of Source Rows to Destination Rows**

  The alignment of the leading pixels in a source row with respect to a destination row influences PIXBLT transfer timing. This alignment determines whether one or two words are required from the source array to fully write the first word of the destination array. This initial condition can be divided into two cases:

  D≥S The four LSBs of the destination address are greater than the four LSBs of the source address. This implies that the amount of data available from the first word of the source array exceeds the amount needed to write to the first word of the destination array. The write to the destination array can proceed immediately.

  D<S The four LSBs of the destination address are less than the four LSBs of the source address. This implies that the amount of data to be written to the first word of the destination array exceeds the amount available from the first word of the source array. Another word must be read from the source array.

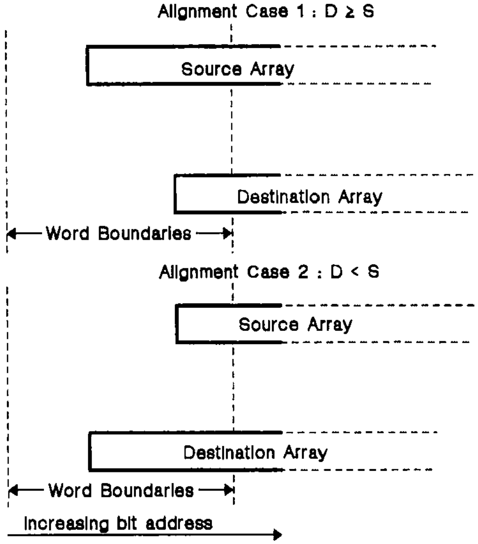

Figure 13-8. Source to Destination Alignments

- **Transfer Direction in X (PBH)**

  PIXBLT transfers proceed a word of data at a time in a consistent direction in X and Y. The default direction is from the smallest word address to the largest, corresponding to left-to-right and top-to-bottom for the default screen orientation. The values of the PBH and PBV bits determine the transfer direction in X and Y.

For the four regular PIXBLTs (without expand), PBH determines the order in which ***words*** are written on each row of the destination array:

PBH=0: Words within rows are written in the order of increasing addresses.

PBH=1: Words are written in the order of decreasing addresses. The value of PBH influences the per-row transfer timings of these PIXBLTs.

The sense of the PBV bit determines the order in which ***rows*** are transferred to the destination array.

PBV=0: Rows are transferred in the order of increasing addresses.

PBV=1: Rows are transferred in the order of decreasing addresses.

This value affects the setup timing, but not the transfer timing.

- **Selected Graphics Operations (*G*)**

Graphics operations such as plane masking, transparency, and pixel processing influence PIXBLT transfer timing because the destination pixels must be read before they are replaced. However, the effects of these operations vary because they are performed by different portions of the TMS34010 hardware. For instance, plane masking, transparency, and field insertion are all performed by the GSP memory controller hardware; any combination of these operations uses 2 machine states for each word written. Pixel processing, on the other hand, is performed by the TMS34010 CPU, and requires 2, 4, 5, or 6 states per word independent, of other operations. *The minimum time for any graphics operation*, then, is **2 machine states**(one memory cycle) using the *replace* operation with plane masking and transparency disabled. These values are shown in Table 13-10.

Table 13-10. Timing Values per Word for Graphics Operations (G)

	Pixel Processing Operation			
Graphics Operation	**Replace**	**Other Booleans or ADD**	**ADDS,SUB MAX or MIN**	**SUBS**
No plane masking or transparency	2	4	5	6
Read-modify-write, plane masking, or transparency	4	6	7	8

- **Alignment/Graphics Adjustment**

An additional adjustment may be necessary when plane masking or transparency are enabled and the alignment type is B, C, or D. As the second line of Table 13-10 shows, if a particular word in a destination row has already been read as part of a read-modify-write operation, no **additional** states are required to perform plane masking or transparency for that word. Since the alignment types with misaligned edges (B, C, and D) already assume a RMW (read-modify-write) on their respective edges, the effect of plane masking or transparency can be ignored for these edges. That is, after you have computed the timing using the proper value for the graphics operation, you can **subtract** 2 states (case B and C) or 4 states (case D) per row from the row timings for the respective alignment cases. Case A requires no adjustment.

13.4.3 PIXBLT Timing Examples

PIXBLT timing is calculated by adding the PIXBLT setup value to the PIXBLT transfer value:

PIXBLT time = PIXBLT setup time + PIXBLT transfer time
- alignment adjustment

PIXBLT setup timings, transfer timings, and the effects of graphics operations are in the following tables:

Table — Page
13-8 PIXBLT Setup Time 13-16
13-9 PIXBLT Transfer Timing† 13-18
13-10 Timing Values per Word for Graphics Operations (G) 13-22

The following three examples illustrate timing for a **PIXBLT XY,L** with these initial implied operand values:

SADDR	=	>003A00E6	(X=230, Y= 58)
SPTCH	=	>800	(X extent = 512 pixels × 4 bits per pixel)
DADDR	=	000030E8	(linear address)
DPTCH	=	>800	(X extent = 512 pixels)
OFFSET	=	00040000	
WSTART	=	00000000	(ignored)
WEND	=	01000100	(ignored)
DYDX	=	000F0036	(DY=15, DX=54)
PSIZE	=	>4	
CONVSP	=	>14	
CONVDP	=	>14	(ignored)
PMASK	=	>0000	
PBH=1,PBV=1			

The setup and transfer timings for these examples are the same, except each uses a different graphics operation. Figure 13-9 illustrates the destination array and window used in these examples. The shaded portion is the destination array.

- **Setup Time:** Windowing is not enabled for this example. The starting corner must be adjusted in both the X and Y dimensions. As Table 13-8 shows, the setup time for a PIXBLT XY,L with these options is 9 + 3 machine states.

- **Transfer Time:** The source and destination arrays have the same pitch; the X and Y dimensions are the same. The Y dimension is 15, so L=15. The X dimension is 54 pixels, and the pixel size is four; 54 divided by 4 produces 13.5, so N=14. N is greater than 3, so this example conforms to the long case. The four LSBs of DADDR are greater than the four LSBs of SADDR ($D \geq S$). The trailing edge is word aligned but the leading edge is not, so the alignment type is C. As Table 13-9 shows, the transfer time for this PIXBLT XY,L with PBH=1 is $[5+(2+G)N]L+5$.

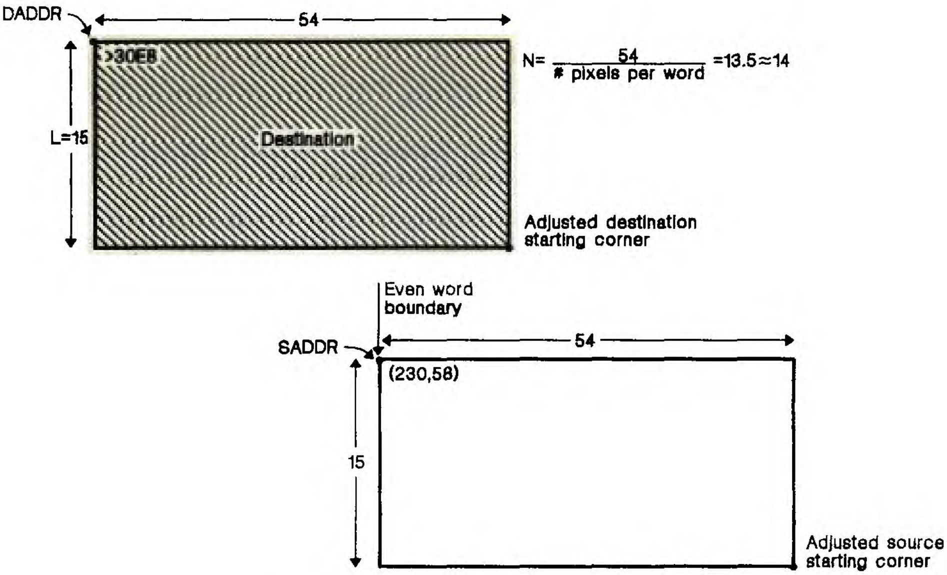
DADDR
54
>30E8
L=15
Destination
N= 54 / # pixels per word =13.5≈14
Adjusted destination starting corner
Even word boundary
SADDR
54
(230,58)
15
Adjusted source starting corner

Figure 13-9. PIXBLT XY,L Timing Example

Example 13-4. W=0, T=0, PP=0, No Plane Masking

The pixel processing *replace* operation has been selected, and transparency and plane masking are not enabled. According to Table 13-10, $G=4$. The total machine states required for this instruction are:

$$\begin{aligned}
\text{PIXBLT time} &= \text{PIXBLT setup time} &&+ \text{PIXBLT transfer time}\\
&= 9 + 4 &&+ [5+(2+G)N]L + 5\\
&= 13 &&+ (5 + 4\times 14)\times 15 + 5\\
&= 920 \text{ states}
\end{aligned}$$

920 states are needed to read and write 810 pixels (at four bits per pixel) with transparency, plane masking, and pixel processing at their default values.

Example 13-5. W=0, T=0, PP=20, No Plane Masking

The pixel processing MAX operation has been selected, and transparency and plane masking are not enabled. According to Table 13-10, $G=5$. Thus, the timing equation becomes:

$$\begin{aligned}
\text{PIXBLT time} &= \text{PIXBLT setup time} &&+ \text{PIXBLT transfer time}\\
&= 9 + 4 &&+ [5+(2+G)N]L + 5\\
&= 13 &&+ (5 + 7\times 14)\times 15 + 5\\
&= 1564 \text{ states}
\end{aligned}$$

1564 states are needed to write the MAX of the pixel values in the source array with those in the destination array for 810 pixels (at four bits per pixel). Transparency and plane masking are at their default values.

Example 13-6. W=0, T=1, PP=5, Plane Masking

The pixel processing XNOR operation has been selected, and transparency and plane masking **are** enabled. According to Table 13-10, G=5. Alignment type C incurs a read-modify-write at the leading edge of each row. The extra read included in the RMW can be used by the plane masking or transparency hardware, so an alignment/graphics adjustment is necessary. The adjustment negates the effect of the extra read cycles in each row that are attributed to the graphics operations. For this example, the amount subtracted is 2 (the number of machine states for a read cycle) times L (the number of rows). The timing is now calculated as:

$$\begin{aligned}\text{PIXBLT time} &= \text{PIXBLT setup time} + \text{PIXBLT transfer time} - \text{adjustment}\\ &= 9 + 4 \quad + [5+(2+G)N]L + 5 \quad - 2L\\ &= 13 \quad + (5 + 8\times14)\times15 + 5 - (2\times15)\\ &= 1743 \text{ states}\end{aligned}$$

1743 states are needed to write the XNOR of the pixel values in the source array with those in the destination array for 810 pixels (at four bits per pixel) with both PMASK and T set.

13.4.4 The Effect of Interrupts on PIXBLT Instructions

The PIXBLT instruction may be interrupted on a destination word boundary during the transfer portion of the algorithm. It may also be interrupted at the end of any row in the array. The context of the PIXBLT is saved in reserved registers. The PBX bit is set in the copy of the ST register that is pushed to the stack. The worst case latency caused by an interrupt is 20 machine states for the interrupt to be recognized. The time for the context switch must be added to this; see Section 8.4.1, Interrupt Latency (page 8-5) for context switch timing.

13.5 PIXBLT Expand Instructions

PIXBLT expand instructions include:

- PIXBLT B,L
- PIXBLT B,XY

The total PIXBLT instruction timing is obtained by adding a setup time to a transfer time:

PIXBLT time = PIXBLT setup time + PIXBLT transfer time

- The **setup sequence** executes an initialization sequence, performing any necessary setup operations and translations. (This includes XY-to-linear conversion and window preclipping.) The result of the setup includes the dimensions of the source array.
- The **transfer sequence** performs the actual data transfer from the source array to the destination array.

PIXBLT setup and transfer timings are in the following tables:

Table **Page**
13-11 PIXBLT Expand Setup Time 13-26
13-12 PIXBLT Expand Transfer Timing† 13-27

13.5.1 PIXBLT Setup Time

PIXBLT setup time is the overhead incurred by the PIXBLT instructions from performing initialization, XY conversions, and window operations.

Window operations are performed before the PIXBLT transfer begins. Window options that affect PIXBLT setup timing include:

- No window checking (*W=0*)
- A window clip that requires no change (*array fits*)
- A window clip that affects the starting pointer (*adjust start*)
- A window clip that affects the array transfer dimensions (*dimension adjust*)
- A window clip that affects both the starting and ending pointers (*adjust both*)
- A window *miss* that requests an interrupt
- A window *hit*

Table 13-11 shows the effect of these options on the PIXBLT setup time. Corner adjust operations have no effect on PIXBLT setup timing.

Table 13-11. PIXBLT Expand Setup Time

	Window Operation							Corner Adjust		
Instruction	W=0	Array Fits	Start Adjust	Dimens Adjust	Adjust Both	Miss	Hit	PBH=1 PBV=0	PBH=0 PBV=1	PBH=1 PBV=1
PIXBLT B,L	4	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
PIXBLT B,XY	6	9	17	12	21	N/A	N/A	N/A	N/A	N/A

For example, a PIXBLT B,XY with the preclipping option requiring an adjustment to the end corner of the array requires 12 states of setup time.

13.5.2 PIXBLT Transfer Timing

Table 13-12 shows transfer timing for PIXBLT expand instructions. Transfer timing is the time required (in addition to the setup time) to execute the actual data transfer to memory. Transfer timing is affected by several factors, including the number of rows in the adjusted array (*L*), the number of words affected per row (*N*), graphics operations (*G*), the four possible destination array alignments (A, B, C, and D), and the arrangement of words in source rows. These factors are described in the list that follows the table.

Table 13-12. PIXBLT Expand Transfer Timing†

Destination Alignment	Transfer Timing
Short case	$(3+2R+G)L + 3$
Medium case	
Alignment A or C	$(3+2R+NG)L + 3$
Alignment B or D	$(5+2R+NG)L + 3$
Long case	
Alignment A	$[(3+2R+2GP)S + 2V + NG]L + 3$
Alignment D	$[(7+2R+2GP)S + 2 + 2V + NG]L + 3$

† Subtract any alignment/graphics adjustment from these values

Key:

L Number of rows in the array (below)
N Number of destination words per row (see page 13-27)
R Number of source words involved in set (see page 13-27)
S Number of 32-bit sets in long source rows (DX/32; see page 13-29)
V Number of source words involved in reading source pixels at end of row after all the complete 32-bit sets have been transferred (see page 13-29)
P Current pixel size
G Value dependent on selected graphics operations (see Table 13-13)

- **Number of Rows in the Array (*L*)**

  The working dimensions (*L* rows × *N* words) for the block transfer are determined by the original destination pointer (DADDR) and dimensions (DYDX) in conjunction with window preclipping. The symbol *L* is used to represent the number of rows in the clipped destination array.

- **Alignment of Leading and Trailing words in Rows**

  After clipping, the data transfer portion of the PIXBLT treats the array as a series of *L* rows of *M.* pixels. These *R* pixels are spread across *N* words in each row of the destination array. *N* and *L* affect the transfer timing. This alignment does not vary from row to row because DPTCH is constrained to be a multiple of 16 for binary PIXBLTs.

Figure 13-10 illustrates a single row of a destination array in memory. The PIXBLT algorithm resolves rows into three portions:

1) The leading edge at the beginning of the row
2) The center *N*-2 words of the row
3) The trailing edge at the end of the row

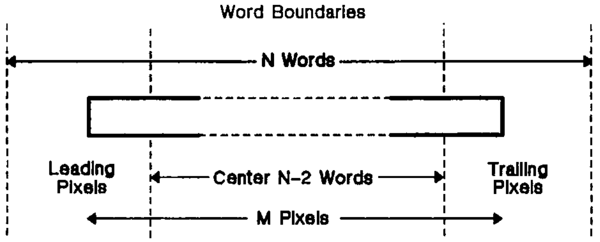

Figure 13-10. Pixel Block Alignment in X

As Figure 13-10 shows, a row of *N* words includes one word each for the leading and trailing parts of the transfer and *N*-2 words for the center portion. PIXBLT expand instructions always transfer the center portion of the row as a series of 16 bit words, and are not affected by the alignment of the leading word. Thus, the alignment of the trailing words in the row characterize the alignment type for the row. Figure 13-11 illustrates the four possible alignments (A, B, C, and D) of a row in the destination array.

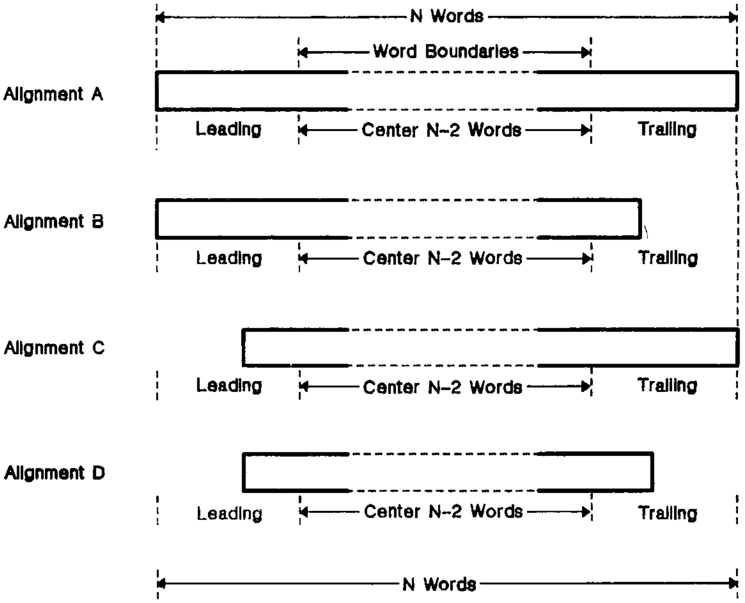

Figure 13-11. Pixel Block Row Alignments

- **Row Length (*N* Words per Row)**

  Row length is determined by a combination of the computed array pointer value in DADDR, the clipped DX dimension, and the pixel size stored in the PSIZE register. The data transfer algorithm breaks down into one of three cases, short, medium, or long, according to the number of words *N* in a row. These three cases include:

  > **Short case**. A row of source array pixels is contained in 16 bits or less and the expanded data involves only one word of the destination array per row ($N=1$). Alignment does not affect the short case.
  >
  > **Medium case**. A row of source array pixels is contained in 32 bits or less but the expanded data involves more than one word of the destination array per row ($N>1$). In this case, the`array alignment is determined by the alignments of the last word in the row. Thus, alignments A,C and B,D have equal transfer timings.
  >
  > **Long case**. A row of source array pixels is contained in more than 32 bits. The expanded data involves multiple words in the destination array row. In this case, the array alignment is determined by the alignments of the last word in the row. Thus, alignments A and B and alignments C and D have equal transfer timings.

  Note that the timings for the short and medium row lengths are not affected by the alignment of the first word on each row of the destination array. That is, the destination array row transfer can start with either a write or a read-modify-write. The long case is treated as a series of 32-pixel medium cases followed by a short case (if necessary) at the end of each row. Each 32-pixel set is expanded and written to the destination in a serial fashion, without optimizing for beginning and ending alignments. Thus, the timing for the long case becomes a product of the number of 32-pixel sets (*S*) and the timing for each set, plus the timing for expanding any remaining segment of the source array (less than or equal to 32 bits) that is left in the row. Note that the remaining segment of the source array may have an alignment type (B or C) that is different from the preceding 32-bit sets.

- **Arrangement of Source Rows**

  As discussed in the *Row Length* section, the number of bits in a row of the source array affects the time required to perform the PIXBLT transfer algorithm. The short and medium cases have explicit timings based on the number of words read from the source row, *R*. Note that the timings for the short and medium row lengths are not affected by the alignment of the last word on each row of the destination array. That is, the destination array row transfer can either end with a write or a read-modify-write.

  The long case is treated as a series of 32-pixel segments followed by a partial segment if necessary at the end of each row. Each 32-pixel set is expanded and written to the destination in a serial fashion without optimizing for beginning and ending alignments. Thus, the timing for the long case becomes a product of the number of 32-pixel sets (*S*) and the timing for each set plus the timing for expanding any remaining segment of the source array (less than 32 bits) that is left in the row. Note that the remaining segment of the source array has an alignment type that is related to the alignment of the preceding 32-bit sets.

The PIXBLT does not attempt to optimize read operations from the source array; therefore, depending on the alignment of the source array, either two or three words may need to be read in order to obtain a 32-bit set of source pixels for expansion. This value, *R*, is the number of source words involved in a 32-bit set of source pixels and may be either two or three. The timings is affected by *R* as wells as the number of such complete 32-bit sets *S* in a source row.

The bits remaining after all of the complete 32-bit sets have been transferred using an abbreviated portion of the long case. Depending on the number of remaining bits and the alignment of the source array, either one, two, or three words may need to be read in order to obtain the remaining set of source pixels for expansion. This value, *V*, for the remaining bits is the number of source words involved while *N* is the number of destination words involved for this fragment.

- **Transfer Direction in X (PBH Bit)**

  These PIXBLT instructions proceed a single word of pixels at a time in the direction of increasing X and increasing Y. This corresponds to left-to-right and top-to-bottom for the default screen orientation. Setting the PBH and PBV bits has no effect.

- **Selected Graphics Operations (*G*)**

  Graphics operations such as plane masking, transparency, and pixel processing influence PIXBLT transfer timing because the destination pixels must be read before they are replaced. However, the effects of these operations are performed by different parts of the TMS34010 hardware. For instance, plane masking, transparency, and field insertion are all performed by the GSP memory controller hardware; any combination of these operations uses 2 machine states for each word written. Pixel processing, on the other hand, is performed by the TMS34010 CPU, and requires 2, 4, 5, or 6 states per word independent of other operations. *The minimum time for any graphics operation*, then, is **2 machine states** (one memory cycle) using the replace operation with plane masking and transparency disabled. These values are shown in Table 13-13.

Table 13-13. Timing Values per Word for Graphics Operations (G)

	Pixel Processing Operation			
Graphics Operation	Replace	Other Booleans or ADD	ADDS,SUB MAX or MIN	SUBS
No plane masking or transparency	2	4	5	6
Read-modify-write, plane masking, or transparency	4	6	7	8

- **Alignment/Graphics Adjustment**

  An additional adjustment may be necessary when plane masking or transparency are enabled and the alignment type is B, C, or D. As the second line of Table 13-13 shows, if a particular word in a destination row has already been read as part of a read-modify-write operation, no **additional** states are required to perform plane masking or transparency for that word. Since the alignment types with misaligned edges (B, C, and D) already assume a RMW (read-modify-write) on their respective edges, the effect of plane masking or transparency can be ignored for these edges. That is, after you have calculated the timing using the proper value for the graphics operation, you can **subtract** 2 states (cases B and C) or 4 states (case D) per row from the row timings for the respective alignment cases. Case A requires no adjustment.

13.5.3 PIXBLT Timing Examples

PIXBLT timing is calculated by adding the PIXBLT setup value to the PIXBLT transfer value:

PIXBLT time = PIXBLT setup time + PIXBLT transfer time
- alignment adjustment

PIXBLT setup timings, transfer timings, and the effects of graphics operations are listed in the following tables:

Table		Page
13-11	PIXBLT Expand Setup Time	13-26
13-12	PIXBLT Expand Transfer Timing†	13-27
13-13	Timing Values per Word for Graphics Operations (G)	13-30

The following three examples illustrate timing for a **PIXBLT B,XY** with these initial implied operand values:

SADDR	=	>0003E2E8	(linear address)
SPTCH	=	>00AD0	(X extent = 2768 pixels)
DADDR	=	>0032010B	(X=267, Y= 50)
DPTCH	=	>800	(X extent = 512 pixels)
OFFSET	=	>00040000	
WSTART	=	>00000000	(ignored)
WEND	=	>01000100	(ignored)
DYDX	=	>000A000A	(DX=10, DY=10)
PSIZE	=	>8	
CONVSP	=	>xxx	(ignored)
CONVDP	=	>14	
PMASK	=	>0000	
W=0,T=0,PP=0			
PBH=1,PBV=1			(ignored)

This PIXBLT B,XY examples expand a 10-by-10 font (*L*=10) into eight bits per pixel with color. The setup and transfer timings for these examples are the same, except each uses a different graphics operation. Figure 13-12 illustrates the destination array and window used in these examples. The shaded portion is the destination array.

- **Setup Time:** Windowing is not enabled for this example. PBH and PBV are ignored. As Table 13-11 shows, the setup time for a PIXBLT XY,L with these options is 6 machine states.

- **Transfer Time:** The source is part of a packed font. The source array starts in the middle of a word and extends into the next word, so two words are read for each row of the font (*R*=2). The Y dimension is 10 (*L*=10). Neither the leading nor the trailing edges are word aligned, so the alignment type is D. The X dimension is 10 pixels wide, but with alignment type D, an extra word is involved for both the leading and trailing pixels; the pixel size is eight, so 12 divided by 2 (two pixels per word) produces *N*=6. Since the width is less than 32 pixels (10), but more than one word of the destination is affected, this example is a medium case. As Table 13-12 shows, the transfer timing is (5+2*R*+2*GN*)*L* + 3.

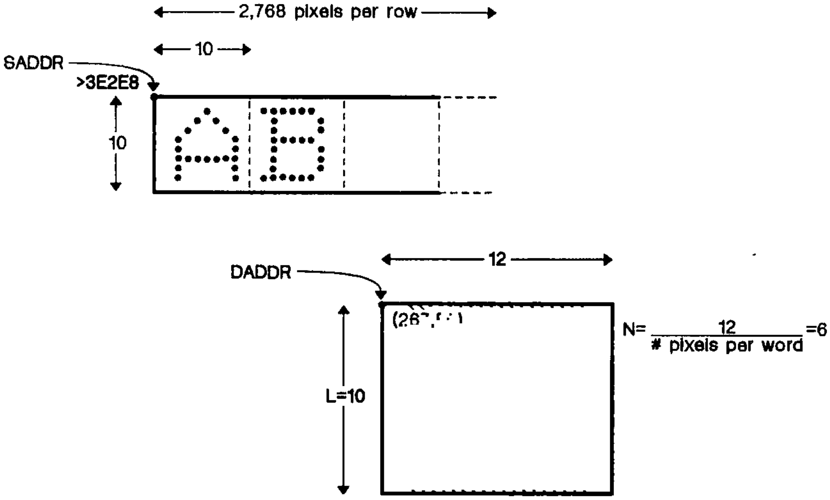

Figure 13-12. PIXBLT B,XY Timing Example

Example 13-7. W=0, T=0, PP=0, No Plane Masking

The pixel processing *replace* operation has been selected, and transparency and plane masking are not enabled. According to Table 13-13, ***G*=2**. The total machine states required for this instruction are:

PIXBLT time	=	PIXBLT setup time	+	PIXBLT transfer time
	=	6	+	(5+2*R*+*NG*)*L* + 3
	=	6	+	(5 + 2x2 + 6x2) x10 + 3
	=	219 states		

219 states are needed to read, expand, and write 100 pixels (at eight bits per pixel) with transparency, plane masking, and pixel processing are at their default values.

Example 13-8. W=0, T=0, PP=20, No Plane Masking

The pixel processing MAX operation has been selected, and transparency and plane masking are not enabled. According to Table 13-13, ***G*=5**. Thus, the timing equation becomes:

$$\begin{aligned}\text{PIXBLT time} &= \text{PIXBLT setup time} &&+ \text{PIXBLT transfer time}\\ &= 6 &&+ (5+2R+NG)L + 3\\ &= 6 &&+ (5 + 2\times2 + 6\times5)\times10 + 3\\ &= 399 \text{ states}\end{aligned}$$

399 states are needed to read, expand, and write 100 pixels (at eight bits per pixel) using the MAX operator with transparency and plane masking at their default values.

Example 13-9. W=0, T=1, PP=5, Plane Masking Enabled

The pixel processing XNOR operation has been selected, and transparency and plane masking **are** enabled. According to Table 13-13, ***G*=6**. Alignment type D incurs a read-modify-write at the leading and trailing edges of each row. The extra read included in the RMW can be used by the plane masking or transparency hardware, so an alignment/graphics adjustment is necessary. The adjustment negates the effect of the extra read cycles in each row that are attributed to the graphics operations. For this example, the amount subtracted is 4 (the number of machine states for a read cycle times 2) times *L* (the number of rows). The timing is now calculated as:

$$\begin{aligned}\text{PIXBLT time} &= \text{PIXBLT setup time} &&+ \text{PIXBLT transfer time} &&- \text{adjustment}\\ &= 6 &&+ (5+2R+NG)L + 3 &&- 4L\\ &= 6 &&+ (5 + 2\times2 + 6\times6)\times10+3 &&- (4\times10)\\ &= 419 \text{ states}\end{aligned}$$

419 states are needed to read, expand, and write 100 pixels (at eight bits per pixel) using the XNOR operator with transparency and plane masking active.

13.5.4 The Effect of Interrupts

The PIXBLT instruction may be interrupted on a destination word boundary during the transfer portion of the algorithm. It may also be interrupted at the end of any row in the array. The context of the PIXBLT is saved in reserved registers. The PBX bit is set in the copy of the ST register that is pushed to the stack. The worst case latency caused by an interrupt is 20 machine states for the interrupt to be recognized. The time for the context switch must be added to this; see Section 8.4.1, Interrupt Latency (page 8-5) for context switch timings.

13.6 The LINE Instruction

The total LINE instruction timing is obtained by adding a setup time to a transfer time:

LINE time = LINE setup time + LINE transfer time

- The **setup sequence** executes an initialization sequence, performing any necessary setup operations and translations.
- The **transfer sequence** performs the actual data transfer from the source array to the destination array.

13.6.1 LINE Setup Time

LINE setup time is the overhead incurred from initiating the LINE instruction. ***It is always 4 machine states.***

13.6.2 LINE Transfer Timing

Table 13-14 shows LINE transfer timing. LINE transfer timing may be influenced by window and pixel processing operations; their affects are discussed in the list that follows Table 13-14.

Table 13-14. LINE Transfer Timing

	Window Option			
Instruction	W=0 (Off)	W=1 Window Hit	W=2, Interrupt On Clip	W=3 Clipping
LINE 0	$(3+P)E$	$(3+P)E + 5Q$	$(3+P)E$ †	$5Q + 5$
LINE 1	$(3+P)E$	$(3+P)E + 5Q$	$(3+P)E$ †	$5Q + 5$

† Add 5 for a window violation

Key:

E Number of pixels written
Q Number of pixels calculated, but not written
P Selected pixel processing operation

- **Window Checking**

Although window operations affect the setup time of most instructions, they are performed *during transfer execution* of the LINE instruction, affecting it on a per-pixel basis. Window operations that affect the LINE instruction include:

- No window checking
- Window clip: V flag set, LINE aborted on first write outside window
- Window hit: WVP flag set, V flag cleared, abort LINE on first write inside window

- **Pixel Processing Operations**

Pixel processing operations influence the LINE transfer timing. (The effects of other graphics operations, such as plane masking and transparency, are already included.) Pixel processing consumes 2, 3, 4, or 5 machine states per pixel, depending on the operation selected. Table 13-15 shows the effects of pixel processing on LINE timing.

Table 13-15. Per-Word Timing Values for Pixel Processing (P)

Replace	Other Booleans or ADD	ADDS,SUBS MAX or MIN	SUBS
2	4	5	6

13.6.3 LINE Timing Example

This example illustrates timing for a **LINE 0**, drawing a line from (3,52) to (19,55). Assume the following registers have been loaded with these values:

B0	>FFFF FFF1	Decision variable $d = 2b - a = -15$
B2	= >0052 0003	DADDR
B3	= >0000 0800	DPTCH (CONVDP=13)
B4	= >0000 0100	OFFSET
B5	= >0030 0003	WSTART
B6	= >0055 0025	WEND
B7	= >0003 0016	b:a; b=3 and a=22
B9	= >4444 4444	COLOR1 (color of the line)
B10	= >0000 0017	COUNT (a+1)
B11	= >0001 0001	Diagonal increment (+1,+1)
B12	= >0000 0001	Nondiagonal increment (0,+1)
B13	= >FFFF FFFF	PATTRN (all 1s)

W=3, T=0, PP=0, No plane masking

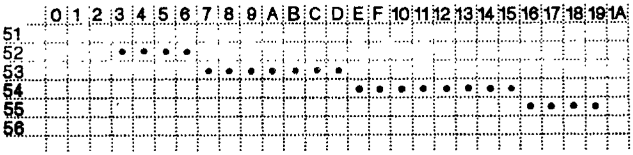

Figure 13-13. LINE Timing Example

- **Setup Time:** The setup time for a LINE instruction is always 4 machine states.

- **Transfer Time:** Windowing is on for this LINE 0 instruction; as Table 13-14 shows, the transfer timing is $(3+P)E + 5Q$. The pixel processing *replace* operation has been selected; according to Table 13-15, $\boldsymbol{P=2}$. Register B10 indicates the number of pixels that will be drawn (E=23). Since the line fits within the window, all pixels calculated are drawn; thus, Q=0.

The total machine states required for this instruction are:

$$\begin{aligned}
\text{LINE time} &= \text{LINE setup time} &&+ \text{LINE transfer time} \\
&= 4 &&+ (3+P)E + 5Q \\
&= 4 &&+ (3+2) \times 23 + 0 \\
&= 119 \text{ states}
\end{aligned}$$

119 states are needed to draw these 23 pixels.

13.6.4 Effects of Interrupts on LINE Timing

The LINE instruction may be interrupted on any pixel boundary during the transfer portion of the algorithm. The context of the LINE is saved in reserved registers; the PC is decremented before it is pushed on the stack, so that execution returns to the LINE opcode. This operation takes 20 machine states for the interrupt to be recognized. The time for the context switch must be added; see the TRAP instruction for context switch timing.