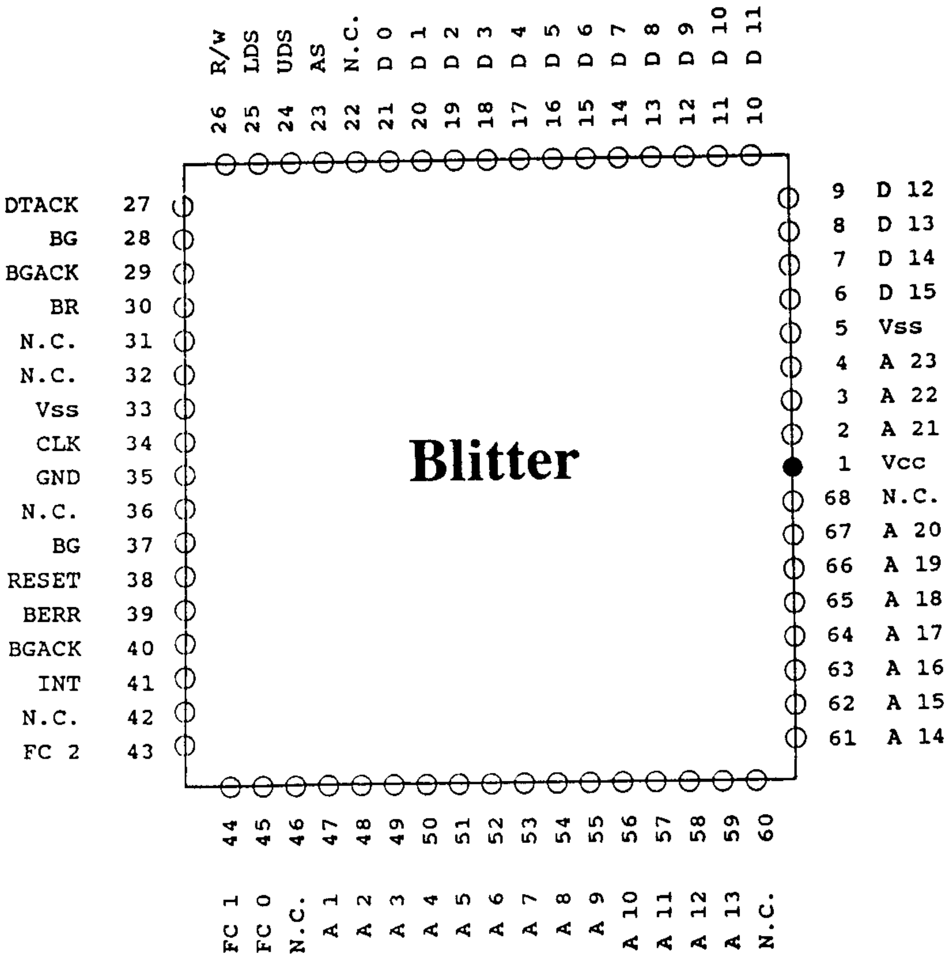
4.3 The blitter chip

Anyone who has followed the development of the ST has surely heard the word *blitter*. More than two years were spent developing the blitter chip. The main advantage of this chip is its speed, working with data in the DMA register. The blitter uses a memory range independent of the 68000 microprocessor. Without the blitter chip, you need several kilobytes of program code to realize graphics through software.

The basic graphic routines of the ST are accessed by software through line-A opcodes. The blitter can take on parts of these routines and execute them faster than the 68000 could handle them. That is first taken by the BITBLT function, shifting the established pixel-oriented memory range. However, the fill can be taken up in any memory range. The details of the blitter options follow later. First let's look at chip design.

Figure 4.3-1 BLITTER



Since the blitter is a DMA device, it must be able to transfer the processor in an idle state. The processor needs the 68000 pins BR (Bus Request), BG (Bus Grant) and BGACK (Bus Grant Acknowledge). The BG pin conveys everything needed for the address and data bus. If the processor recognizes a Bus Request, BG tells the attached device that there is now a bus available for the DMA device. Now a short delay loop executes until the 68000 stops its activity in the different pins (see Section 1.2). As long as the DMA entry has established that the processor is no longer active, then it restarts with the help of BGACK. After data transfer finishes, BGACK clears, and the processor receives control of the bus.

The blitter chip can use the entire address range of the 68000 (16 megabytes). In order to manipulate the data in memory through programming, the processor cannot produce any control signals. These controlled by the READ/WRITE pin, which determines which data is read and which is written to memory. Other important signals for accessing memory are AS (Address Strobe), LDS (Lower Data Strobe) and UDS (Upper Data Strobe).

The DTACK signal (Data Transfer Acknowledge) invokes the blitter chip only, when the processor displays the transfer of data. It cannot do the DMA transfer itself, since the RAM chip timing is set by the blitter or the CLK signal. Like the other onboard DMA channels (floppy disk and DMA port) and the ACIAs, the blitter is also capable of performing interrupts. This means that it can create its own interrupts to end data transfers. Therefore, it uses the free bit 3 of the MFP interrupt entry (GPIP). This option is not usually used by the ST operatng system. However, other interrupt-oriented operating systems like RTOS, OS9 or UNIX should have blitter integration.

The last group of blitter connections belong to the power connections. In addition to the usual 5 volt current and ground, the blitter needs a time signal of 8 mHz.

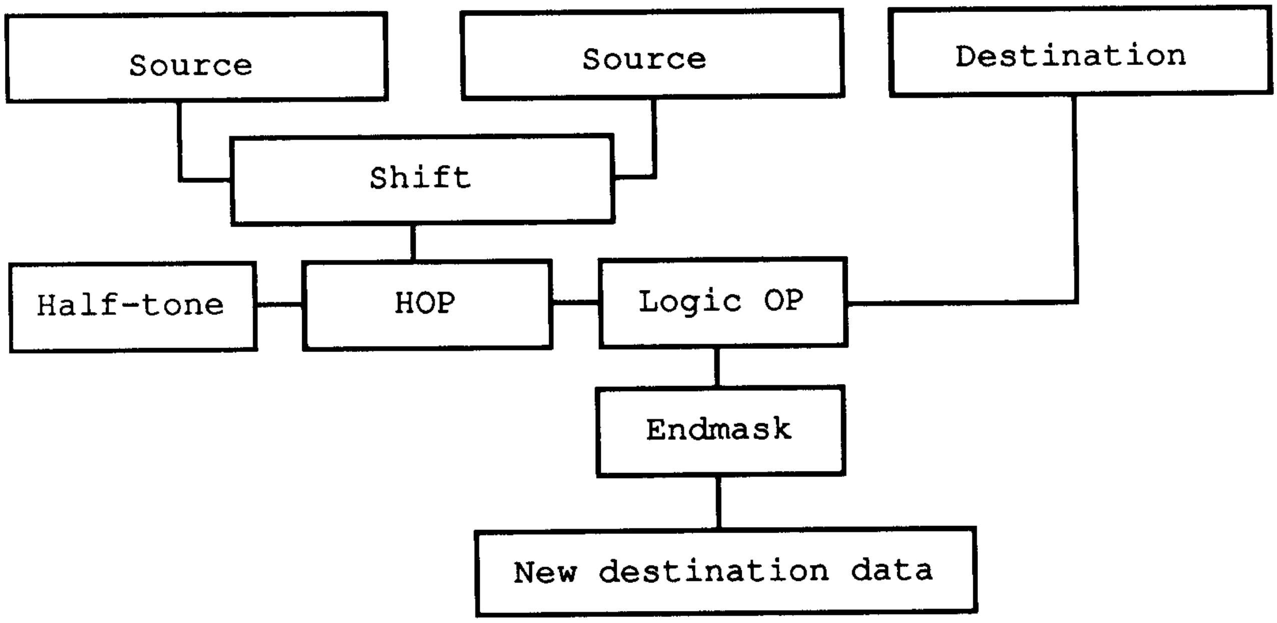
4.3.1 The blitter registers

The ST blitter chip is the hardware implementation of the BITBLT algorithm used in the line-A opcodes.

Figure 4.3.1-1 shows a block diagram of the blitter functions. The blitter can basically set up a source range which can be combined with a current raster, a destination range of 16 different logical operands, and a destination range in which it stores the result. Both source and destination ranges can be stored in the same area of RAM. Unlike the processor, which can only operate in bytes and words, the blitter is bit-oriented. This makes the blitter ideal for handling bitmapped graphics. It is also practical for normal copy and transfer commands, e.g., high-speed RAM disk operations without hard disk interrupts.

The following is a look at the individual registers used by the blitter:

Figure 4.3.1-1 BLITTER BLOCK DIAGRAM



The first 16 registers are marked as half-tone RAM, and contain the raster used in half-tone operations. The registers are each 16 bits wide. When the raster is used, a proportional register for a lin is used. The raster repeats over all 16 lines. The Line Number register (see below) determines which half-tone register is used next.

```
         Bit  F E D C B A 9 8 7 6 5 4 3 2 1 0
$FF8A00  R/W  X X X X X X X X X X X X X X X X Half-tone RAM  0
$FF8A02  R/W  X X X X X X X X X X X X X X X X Half-tone RAM  1
$FF8A04  R/W  X X X X X X X X X X X X X X X X Half-tone RAM  2
$FF8A06  R/W  X X X X X X X X X X X X X X X X Half-tone RAM  3
$FF8A08  R/W  X X X X X X X X X X X X X X X X Half-tone RAM  4
$FF8A0A  R/W  X X X X X X X X X X X X X X X X Half-tone RAM  5
$FF8A0C  R/W  X X X X X X X X X X X X X X X X Half-tone RAM  6
$FF8A0E  R/W  X X X X X X X X X X X X X X X X Half-tone RAM  7
$FF8A10  R/W  X X X X X X X X X X X X X X X X Half-tone RAM  8
$FF8A12  R/W  X X X X X X X X X X X X X X X X Half-tone RAM  9
$FF8A14  R/W  X X X X X X X X X X X X X X X X Half-tone RAM 10
$FF8A16  R/W  X X X X X X X X X X X X X X X X Half-tone RAM 11
$FF8A18  R/W  X X X X X X X X X X X X X X X X Half-tone RAM 12
$FF8A1A  R/W  X X X X X X X X X X X X X X X X Half-tone RAM 13
$FF8A1C  R/W  X X X X X X X X X X X X X X X X Half-tone RAM 14
$FF8A1E  R/W  X X X X X X X X X X X X X X X X Half-tone RAM 15
```

The next register is called X Increment. This is a leading character dependent 15-bit register. The lowest bit is ignored and constantly registers 0. This makes only even numbers possible. The register gives the offset in bytes in the next source word in the same line. Normally, the Atari gives a 2 for monochrome mode. This is also the case when all planes are copied in color mode. If a plane is copied in medium-res or low-res mode,then 4 or 8 must exist in this register.

```
         Bit  F E D C B A 9 8 7 6 5 4 3 2 1 0
$FF8A20  R/W  X X X X X X X X X X X X X X X 0  Source X
                                            |  Increment
                         (always zero, even increments only)
```

The Source Y Increment register determines how many bytes must be added to the current source address, in order to figure out the distance from the end of the current line to the start of the next line.In monochrome mode, a set of pixels measures 80 bytes: When only a segment of 20 bytes is copied, the Source Y Increment gives a value of 60.

```
         Bit  F E D C B A 9 8 7 6 5 4 3 2 1 0
$FF8A22  R/W  X X X X X X X X X X X X X X X 0  Source Y
                                            |  Increment
                         (always zero, even increments only)
```

The Source Address register determines the starting address at the beginning of the copy. It can read or write long word accesses. Bits 0 and 24-31 are used only for even 24- bit addresses. The contents of this register are incremented as part of the operation with the help of the above mentioned increment register (or decremented, depending on the leading character of the increment register). By reading the source address register, the address of the source word used next is received.

```
        Bit  F E D C B A 9 8 7 6 5 4 3 2 1 0
$FF8A24 R/W  - - - - - - - - X X X X X X X 0  Source Address
                                                High   Word
             (unused)      (24-bit addresses only)

        Bit  F E D C B A 9 8 7 6 5 4 3 2 1 0
$FF8A26 R/W  X X X X X X X X X X X X X X X 0  Source Address
                                           |  Low word
                           (always zero, even increments only)
```

The next three registers contain the endmask, which states which bits are changed and which are unchanged.Since the blitter is pixel oriented, but the bus accesses RAM in words, the first and the last word are read as bits. To write 16 bits over the processor bus, the destination word must first read then change the allowable bits, and transfer the result (Read-Modify-Write). Endmask 1 does this for the beginning of a line, endmask 3 applies to the end of a line. Endmask 2 is used by all other words. It is normally set to $FFFF (all bits are altered by it). Thus, a previous reading of the destination word is unnecessary.

```
        Bit  F E D C B A 9 8 7 6 5 4 3 2 1 0
$FF8A28 R/W  X X X X X X X X X X X X X X X X   Endmask 1
$FF8A2A R/W  X X X X X X X X X X X X X X X X   Endmask 2
$FF8A2C R/W  X X X X X X X X X X X X X X X X   Endmask 3
```

The next three registers are Destination X Increment, Destination Y Increment and Destination Address. They have the same uses as the above-mentioned source registers, except that these three apply to the destination.

```
        Bit  F E D C B A 9 8 7 6 5 4 3 2 1 0
$FF8A2E R/W  X X X X X X X X X X X X X X X 0  Destination X
                                           |  Increment
                           (always zero, even increments only)
        Bit  F E D C B A 9 8 7 6 5 4 3 2 1 0
$FF8A30 R/W  X X X X X X X X X X X X X X X 0  Destination Y
                                           |  Increment
                          (always zero, even increments only)

        Bit  F E D C B A 9 8 7 6 5 4 3 2 1 0
$FF8A32 R/W  - - - - - - - - X X X X X X X X Destination
                                             Address High Word
             (unused)      (24-bit addresses only)

        Bit  F E D C B A 9 8 7 6 5 4 3 2 1 0
$FF8A34 R/W  X X X X X X X X X X X X X X X 0 Destination
                                           | Address Low Word
                          (always zero, even increments only)
```

The X Count register informs you how many words are in a destination line. The minimum value is 1; the highest is 65536 ($0000). Reading the register gives the number of values in this line as words are transferred. When the X Count register is loaded with 1, the values in Destination X Increment, as well as Source X Increment, are unused. Since the line after a word is already the end, and the corresponding Y Increment is used direct.

The Y Count register determines the number of lines. The smallest value is again one, and values of zero are interpreted as 65536. Reading this register gives you the number of lines which need copying. After every transferred line, the value decrements by one until it reaches 0, ending the transfer.

```
        Bit  F E D C B A 9 8 7 6 5 4 3 2 1 0
$FF8A36 R/W  X X X X X X X X X X X X X X X X  X-Count
$FF8A38 R/W  X X X X X X X X X X X X X X X X  Y-Count
```

All the abovementioned registers can only be read as words or long words; byte access is not allowed.

The HOP register determines the combination of source and half-tone RAM. The two lowest bits have the following meanings:

HOP	Combination
0	All 1-bits
1	Half-tone RAM
2	Source
3	Source and half-tone RAM

You can therefore determine whether the source can be used unaltered (HOP = 2), whether the half-tone RAM is combined with the logical AND (HOP = 3) or whether only the half-tone RAM is used (HOP =1). This is useful, for example, when filling an area with a raster pattern. Furthermore, it is still possible to fill the destination with 1-bits (HOP = 0). When half-tone RAM is used, another register determines which half-tone registers are used.

```
             Bit    7 6 5 4 3 2 1 0
$FF8A3A      R/W    - - - - - - X X       HOP
                                          Half-tone operation
```

The next register determines the receiver of the new destination value, after logical operations between destination and source. Here are 16 different options in the following table.

(~s&~d) |	(~s&d) |	(s&~d) |	(s&d)	Operation	New destination
0	0	0	0	0	all 0 bits
0	0	0	1	1	source AND destination
0	0	1	0	2	source AND NOT destination
0	0	1	1	3	source
0	1	0	0	4	NOT source AND destination
0	1	0	1	5	destination
0	1	1	0	6	source XOR destination
0	1	1	1	7	source OR destination
1	0	0	0	8	NOT source AND NOT destination
1	0	0	1	9	NOT source XOR destination
1	0	1	0	10	NOT destination
1	0	1	1	11	source OR NOT destination
1	1	0	0	12	NOT source
1	1	0	1	13	NOT source OR destination
1	1	1	0	14	NOT source OR NOT destination
1	1	1	1	15	all 1 bits

The most important operations are the following three (Replace mode, Source replaces and destination), 6 (XOR mode; overlapping of destination and source) and and 7 (OR mode).

```
             Bit    7 6 5 4 3 2 1 0
$FF8A3B      R/W    - - - - X X X X       OP
                                          Logical operation
```

```
            Bit     7 6 5 4 3 2 1 0
$FF8A3C     R/W     X X X - X X X X
                            |_|_|_|_______Line number
                          |_______________Unused
                        |_________________SMUDGE
                      |___________________HOG
                    |_____________________Busy
```

The next register combines several functions. The lowest 4 bits determine which of the 16 half-tone RAM registers are even used. The value is incremented or decremented after a line, depending on the leading character in the Destination Y Register.When the SMUDGE bit is set, the number of the half-tone RAM register is determined by the four lowest bits of the above mentioned source data. The selected half-tone operation (HOP) stays active. This allows special effects.

The next bit in this register determines the method of bus access in the blitter. When the HOG bit clears, the blitter and processor share the same bus. After 64 bus cycles, the blitter stops and the processor takes over the bus for 64 bus cycles. When the HOG bit is set, the processor stops until the blitter finishes its operations. In either case, other DMA devices (floppy and harddisk) have priority over the blitter. The Prefetch mechanism of the 68000 processor lets you bypass HOG mode, so after the start of the blitter the next processor command executes when the blitter is ready.

The BUSY bit is set, initializing all other blitter registers, in order to start the blitter. It waits until the blitter ends its operation. Since the interrupt output mirrors the status of the blitter, blitter operations can be ended by an interrupt taken from the third bit of the GPIP within the MFP 68901.

```
            Bit     7 6 5 4 3 2 1 0
$FF8A3D     R/W     X X - - X X X X
                            |_|_|_|_______SKEW
                        |_|_______________Unused
                      |___________________NFSR
                    |_____________________FXSR
```

The last blitter register also has several functions. The lowest four bits determine the source operand shifts, to protect the destination operations. Since the blitter is bit-oriented, but bus access is word-oriented, the source data must move to set the bit positions of half-tone masks and destination data. Therefore, two source data words are read, shifting the relevant bits for calling in a 16-bit source register (see Figure 4.3.1-1).

FXSR and NFSR are abbreviations for Force eXtra Source Read and No Final Source Read. When the FXSR bit is set, the beginning of each line is read as an additional source word. The NFSR bit is set when the last word of the source line cannot be read. The use of these bits require changes to Source Y Increment and Source Address Register.

Normally you can access the blitter directly through the operating system. When you use the line-A or VDI functions, the operating system can tell whether the function is produced by software or by the blitter (see XBIOS function $64).