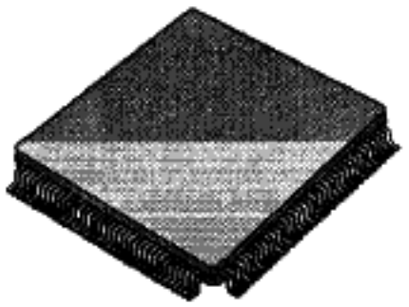
SGS-THOMSON MICROELECTRONICS

STi3220

MOTION ESTIMATION PROCESSOR

- PIXEL RATE FROM 0 UP TO 20MHz
- BLOCK SIZE : 8 x 4n, 16 x 4n
- MAXIMUM DISPLACEMENT +7/-8 PIXELS HORIZONTAL AND VERTICAL
- COMPUTATION OF THE MOTION VECTOR AND MINIMUM DISTORTION
- RANDOM ACCESS TO THE 256 DISTORTIONS
- 8-BIT UNSIGNED INPUT PIXEL
- 4-BIT 2'S COMPLEMENT HORIZONTAL DISPLACEMENT
- 4-BIT 2'S COMPLEMENT VERTICAL DISPLACEMENT
- 16-BIT UNSIGNED DISTORTION VALUES
- CLOCK FREQUENCY = INPUT RATE
- FULLY TTL AND CMOS COMPATIBLE
- CMOS TECHNOLOGY
- SINGLE + 5 VOLT POWER SUPPLY
- MAXIMUM POWER DISSIPATION : 2.4 WATTS AT 20MHz CLOCK RATE



PQFP144
(Plastic Package)

ORDER CODE : STI3220CV

DESCRIPTION

The Real Time Motion Estimation Chip is a dedicated circuit for motion estimation at video rates.

The chip is optimized to compute the displacement vector of 8 x 4n or 16 x 4n blocks in a search window (SW) defined by a maximum horizontal and vertical displacement of +7/-8 pixels corresponding to 256 different vectors.

The chip computes 256 distortions for each block according to the MAE criterion.

The minimum distortion and the corresponding vector are calculated on chip. A random access to all distortions allows to implement more elaborate algorithms at the system level.

Displacement vectors of +15/-16 pixels are accommodated with a single chip for lower rates (time multiplexed) or with several chips for higher rates (spatial multiplexed).

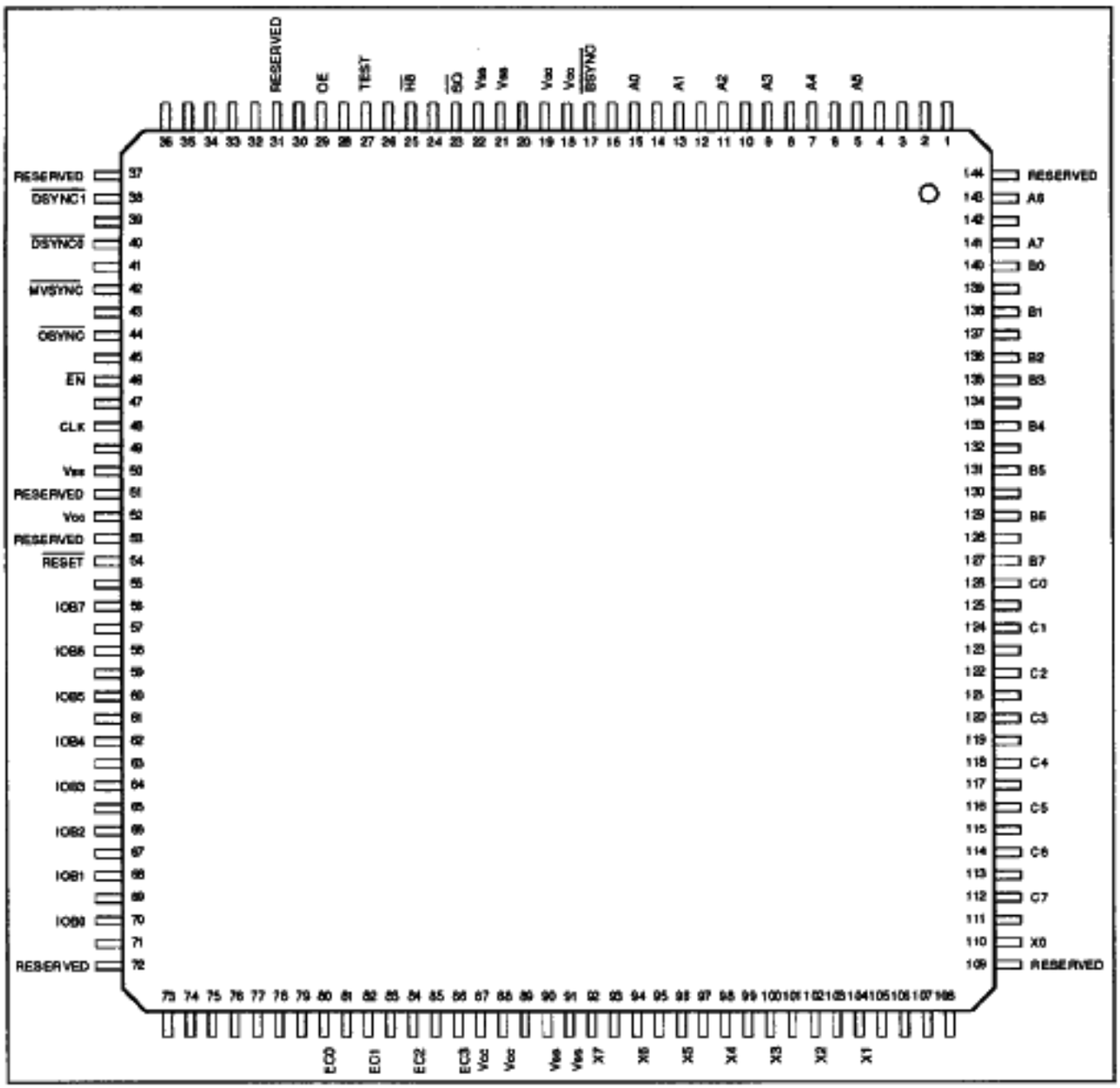
Table 1 shows different application examples for the motion estimation chip.

Table 1

Picture format	Pixel Rate	Block Size	Maximum Displacement	Minimum Chip Clock Frequency	Number of chips
CIF 10Hz	1.01MHz	8 x 4n, 16 x 4n	+7 / -8	1.01MHz	1
			+15 /-16	4.07MHz	1
CIF 30Hz	3.04MHz	8 x 4n, 16 x 4n	+7 / -8	3.04MHz	1
			+15 /-16	12.2MHz	1
TV 25Hz	13.5MHz	8 x 4n, 16 x 4n	+7 / -8	13.5MHz	1
			+15 /-16	13.5MHz	4

3220-01.TBL

PIN CONNECTIONS



1. PIN LIST

Pin Number	IN/OUT	Description
5	IN	A5
7	IN	A4
9	IN	A3
11	IN	A2
13	IN	A1
15	IN	A0
17	IN	$\overline{BSYNC}$
18, 19		V_{CC}
21, 22		V_{SS}
23	IN	$\overline{SQ}$
25	IN	$\overline{H8}$
27	IN	TEST
29	IN	OE
31		Reserved
37		Reserved
38	OUT	$\overline{DSYNC1}$
40	OUT	$\overline{DSYNC0}$
42	OUT	$\overline{MVSYNC}$
44	OUT	$\overline{OSYNC}$
46	IN	$\overline{EN}$
48	IN	CLK
50		V_{SS}
51		Reserved
52		V_{CC}
53		Reserved
54	IN	$\overline{RESET}$
56	IN/OUT	IOB7
58	IN/OUT	IOB6
60	IN/OUT	IOB5
62	IN/OUT	IOB4
64	IN/OUT	IOB3
66	IN/OUT	IOB2
68	IN/OUT	IOB1
70	IN/OUT	IOB0
72		Reserved

Pin Number	IN/OUT	Description
80	IN	EC0
82	IN	EC1
84	IN	EC2
86	IN	EC3
87, 88		V_{CC}
90, 91		V_{SS}
92	IN	X7
94	IN	X6
96	IN	X5
98	IN	X4
100	IN	X3
102	IN	X2
104	IN	X1
109		Reserved
110	IN	X0
112	IN	C7
114	IN	C6
116	IN	C5
118	IN	C4
120	IN	C3
122	IN	C2
124	IN	C1
126	IN	C0
127	IN	B7
129	IN	B6
131	IN	B5
133	IN	B4
135	IN	B3
136	IN	B2
138	IN	B1
140	IN	B0
141	IN	A7
143	IN	A6
144		Reserved

3220-02.TBL

Notes : 1. All other pins are not connected.
2. Test is restricted to SGS-THOMSON Company use (must be grounded).
Signal names are noted with an overbar if they are active low, otherwise they are active high.

2. INTRODUCTION

The STi3220 circuit is intended for use in video compression systems where motion estimation is employed. For example, CCITT H.261 and ISO(MPEG) compatible video compression systems need to perform motion estimation. Potential applications are numerous and include, for example, systems concerned with image transmission and/or storage, e.g videophone, videoconference, digital VTR, multimedia computer, etc.

2.1 Motion Estimation Basics

Motion estimation is one of the techniques that play a role in video picture compression. The basic idea arises from a common sense observation : in a video sequence, successive frames are likely to represent the same details, with little difference between one frame and the next. A sequence showing moving objects over a still background is a good example. Important data compression can be effected if each component of a frame is represented by its difference with the most similar component - the **predictor** - in the previous frame, and by a vector - the **motion vector** - expressing the relative position of the two components. If an actual motion exists between the two frames, the difference may be null or very small. The original component can be reconstructed from the difference, the motion vector, and the previous frame.

Motion estimation is the process of finding a good (if not the best) template for prediction and the corresponding vector. It does not in itself compress data but provides the basic information enabling data compression to be effected.

Several motion estimation techniques exist but the most popular one is based on a block by block processing and for this reason is called **block matching**.

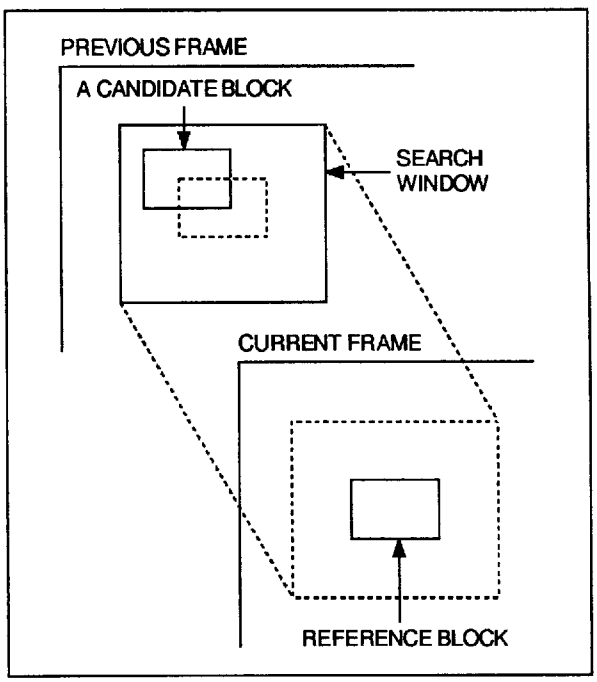
For block matching processing, the frame to be compressed is positioned into blocks which are processed individually. For a given block - the **reference block** - a predictor is searched for among candidate blocks of equal size included in a surrounding rectangular region in the previous frame - the **search window** (see Figure 1). The reference block is compared with the candidate blocks. The result of a comparison is called a **distortion** and serves as a measure of the similarity of the two blocks. The candidate block corresponding to the minimum distortion is the best match, and this block along with the vector relating it to the reference block, form the predictor.

When all the blocks included in the search area are compared with the reference block, the process is called **full search block matching**.

To compare blocks, several criterion exist, among which the **Mean Absolute Error (MAE)**, is the most often used because it offers a good trade-off between complexity and efficiency.

The STi3220 performs full search block matching with the MAE criterion at pixel rates up to 20MHz.

Figure 1 : Block matching notation



2.2 Notation

In Figure 2, the notation used throughout this document is illustrated.

The reference block, referred to as X, is M rows high and N columns wide. A pixel of block X situated at the intersection of row i and column j is denoted Xi,j. For block X, numbering of rows and columns starts at 0.

For the STi3220, M=8 or 16 and N is a multiple of 4 and is denoted by N=4n.

The search window, referred to as Y, is 15 pixels larger than X in each direction, leading to 256 candidate blocks, as many distortions and as many motion vectors. A pixel of search window Y situated at the intersection of row i and column j is denoted by Yi,j. For search window Y, numbering of rows and columns starts at -8.

A motion vector is expressed with two components, a vertical one and a horizontal one. The range of both components is [-8 +7]. Upward and left-hand components are negative; downward and right-hand components are positive.
The set of 256 distortions can be regarded as an array where the indices of a given distortion are precisely the components of the motion vector related to it. Distortion Di,j is related to the motion vector whose components are (i,j) and is equal to :

$$D_{i,j} = \sum_{m=1}^{M} \sum_{n=1}^{N} | X_{m,n} - Y_{m+i,n+j} |$$

3. FUNCTIONALITY

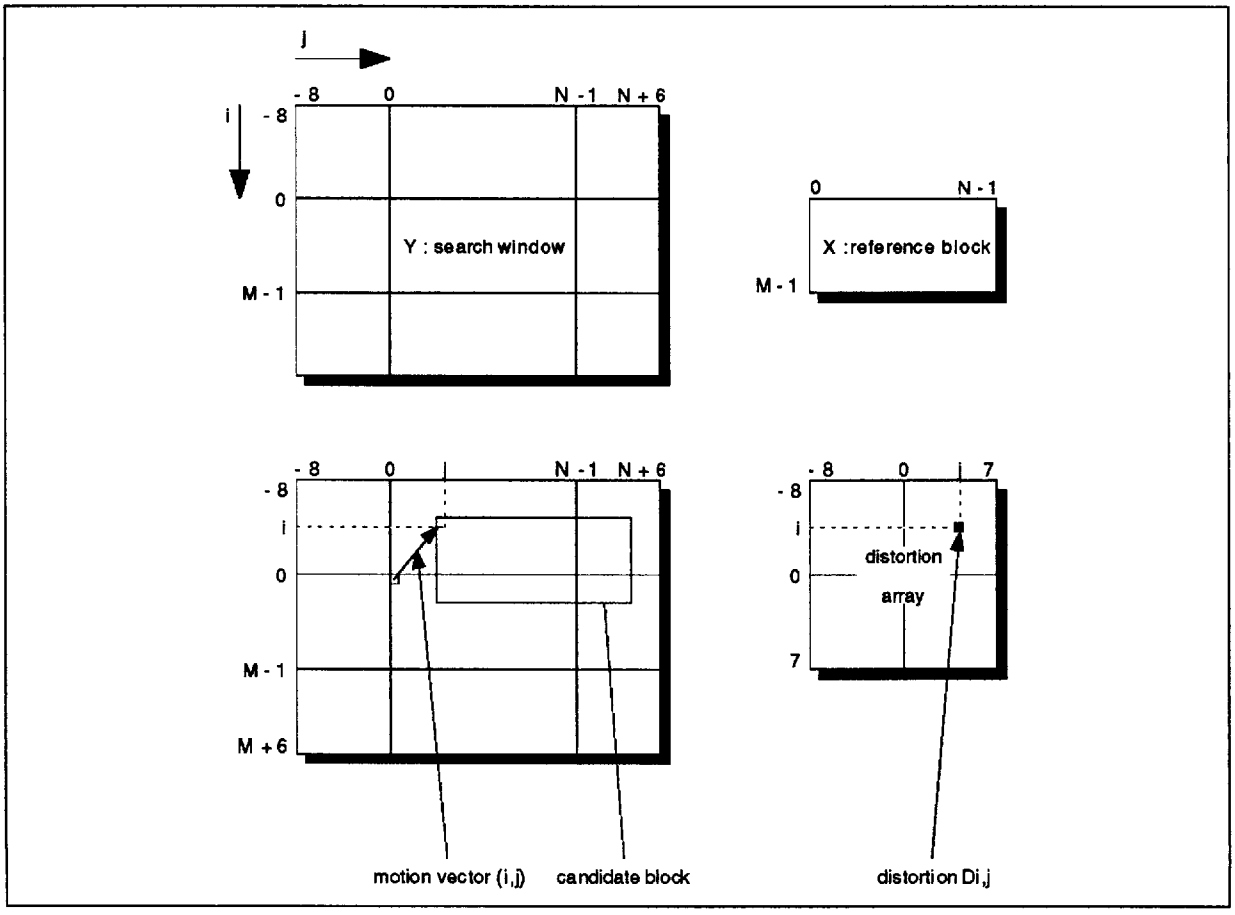
3.1 Overview

The STi3220 contains four 8-bit pixel ports, one to enter the reference block, and the other three to enter the search window. At each clock cycle, one pixel of the reference block and three (in the case of 8xN blocks) or two (in the case of 16xN blocks) pixels of the search window are entered in the circuit. For more details on how to enter pixels for the various block sizes, see section 4.

The STi3220 performs full search block matching with the MAE criterion over a search area expanding 15 pixels around the reference block, 8 pixels in the north and west directions and 7 pixels in the south and east directions. The reason for this asymmetry is that the number of candidates is 256 so that the full range of the 8 bit motion vector can be used. The STi3220 can be used to perform motion estimation over larger search windows by using several circuits in parallel or one circuit in a time multiplex fashion. For more details concerning larger search windows, see section 5.

The STi3220 accommodates various block sizes. Block height is 8 or 16 and block width is a multiple of four. As internal accumulators are 16 bit wide, blocks of more than 256 pixels may cause overflow and generate erroneous results. Thus, the practical maximum sizes of blocks are 8x32 and 16x16. However, wider blocks can still be processed, using 7-bit pixels for example. For more details on how to set the block size, see section 4.9.

Figure 2 : Notation



The STi3220 provides control to manage the various cases when a block is situated near the edge of a picture. In such a situation, the search window is smaller and the motion vector set is reduced. There are 9 different cases depending on which edge/corner of the picture is concerned. For more details on how to set edge control, see section 4.8. For each reference block entered on the circuit, a motion vector and the corresponding minimum distortion are systematically delivered a few cycles after the last pixel of the reference block has been entered in the circuit. When several distortions are equal to the minimum, the circuit outputs the vector which has the highest priority according to a priority table given in section 4.6.

Results are delivered on an 8-bit bidirectional bus. The directionality of this bus is entirely controlled by the system. Motion vectors and minimum distortions are delivered through this bus. Also, the circuit can read in an address and write out the corresponding distortion. For more details on the management of this bus, see sections 4.6 and 4.7.

The STi3220 can be operated in an efficient pipeline mode where successive reference blocks are processed without any dead cycle between the last pixel of a block and the first pixel of the following block. In this mode, the circuit performs motion estimation at the video pixel rate, up to 20MHz. For more details on modes of operation, see section 4.

3.2 Block Diagram

Figure 3 features the block diagram of the circuit.

The STi3220 is composed of a 256 processor systolic array which computes the distortions, a 256x16 SRAM where distortions are temporarily stored, a comparator which computes the minimum among the distortions stored in RAM, a bidirectional bus which is used to input addresses and to output motion vectors and the contents of the RAM, and a set of formatters which reorganize the data coming from the 4 8-bit pixel ports before feeding it to the processors.

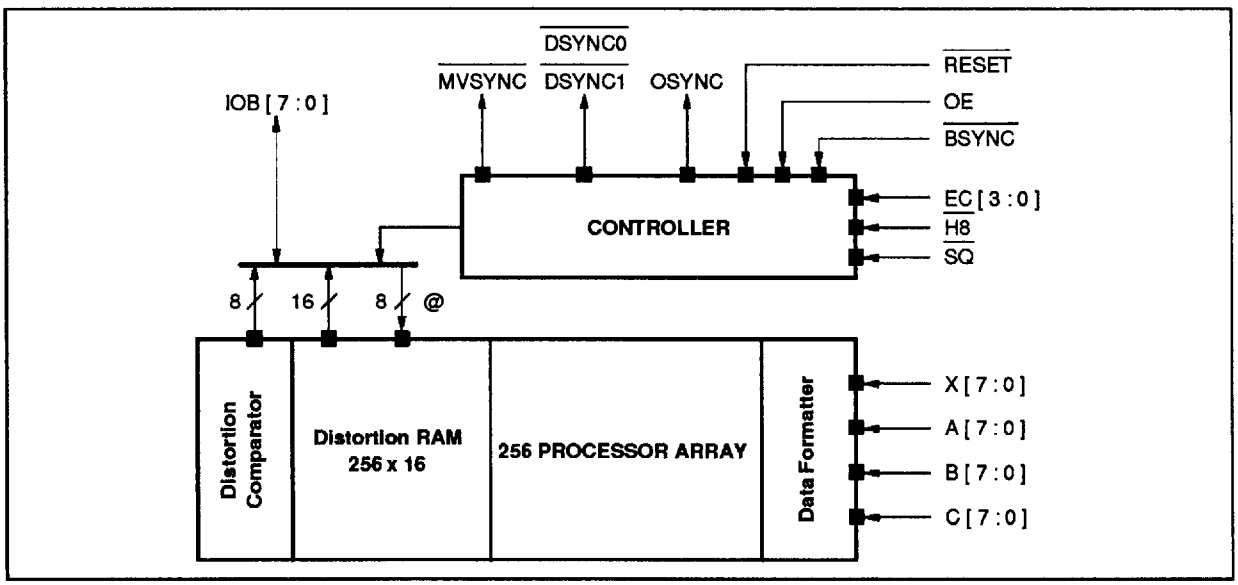
The 256 processors are all identical and work concurrently. Each one is dedicated to one distortion of the distortion array shown above. As soon as a set of computations is finished, the results are simultaneously transferred to the RAM so that a new set of computations can start without any dead cycle.

The RAM can be read from the system at any time, except when a minimum computation is in process. The address of a distortion is precisely the vector related to it.

The comparator is composed of sixteen comparators computing in parallel, so that the 255 comparisons necessary to estimate the minimum distortion are carried out rapidly, in order to set the RAM free as soon as possible for external access. The comparator also stores the address of the minimum which is precisely the motion vector.

The directionality of the I/O bus is entirely controlled by the system through the OE (output enable) pin, so that the system can choose the moment when results are made available.

Figure 3 : STi3220 Block Diagram



4. MODE OF OPERATION

The source format of a digital video signal is usually available in the classical line scanning format. This format is not at all convenient for motion estimation processing. Pixels must be delivered to the STi3220 in the block format which is described hereafter.

Also, when the video signal is a colour video signal, luminance and chrominance pixels are interlaced. Motion estimation is usually performed with the luminance pixels alone, sometimes with the chrominance pixels alone, never with both. This is why in the following, a block of pixels will refer to a block of luminance pixels or a block of chrominance pixels, but not to a block of pixels where chrominance and luminance data is mixed.

Operation of the STi3220 is a succession of sequences of two types: initialization sequences and block sequences. For example, processing one reference block requires an initialization sequence followed by a block sequence. Section 4.4 details the way to combine initialization and block sequences to actually operate the circuit. Sections 4.1 to 4.3 first detail the two types of sequences for different block sizes.

These sequences vary slightly depending on the block height (8 or 16).

During an initialization sequence, for all block sizes, the 15 leftmost columns of the search window are entered into the circuit. No reference block is loaded during this sequence. The duration of an initialization sequence is always 16xM cycles.

During an initialization sequence, the circuit pipeline is initialized with the leftmost part of the search window.

During a block sequence, the N rightmost columns of the search window are loaded into the circuit and the reference block is also loaded into the circuit. The duration of this sequence is always MxN cycles, the time needed to load the reference block.

During a block sequence, the distortions relative to the reference block being loaded are computed.

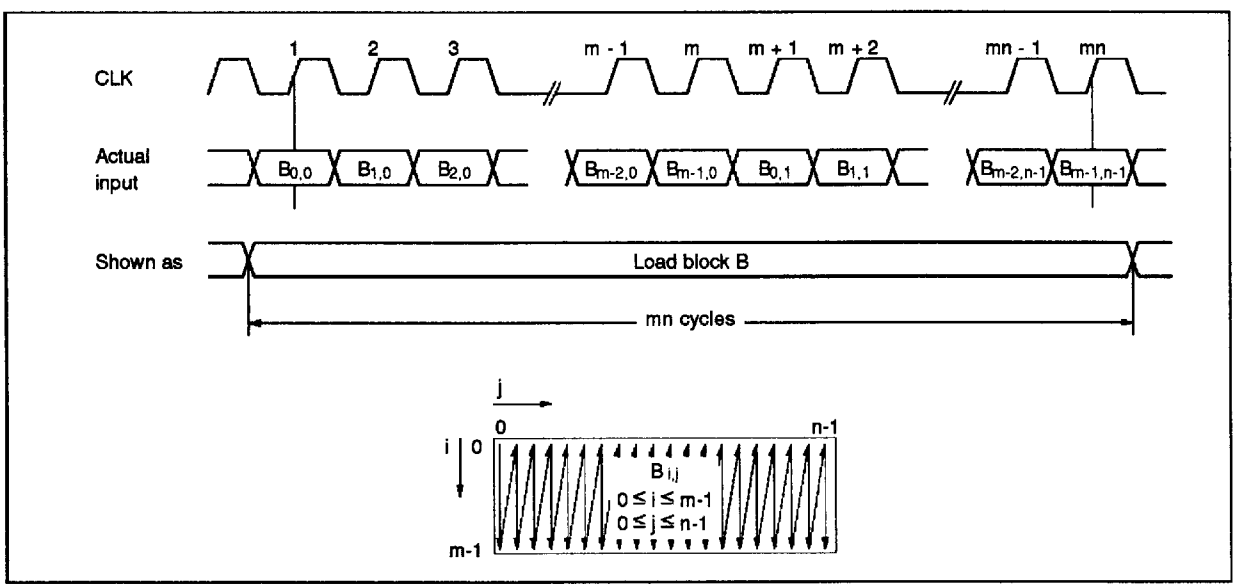
4.1 How to Load a Block

The STi3220 deals with blocks of pixels. A block is a rectangular area of an image and is defined by its width and its height measured in numbers of pixels.

When block B is delivered to the STi3220 (refer to figure 4), it is entered through an 8-bit wide pixel port. The block is scanned column by column, left to right. Each column is in turn scanned from top to bottom. One new pixel of the block must be entered at each clock cycle so that the loading of a mxn sized block will last exactly mxn cycles. Data is latched on the rising edge of CLK. On the rising edge of CLK numbered 1 in figure 4, the first pixel of block B is latched in the circuit.

In all subsequent timing diagrams, whenever a block loading is involved, a simplified representation will be used, as illustrated in Figure 4.

Figure 4 : How to load a block



4.2 8x4n blocks

For 8x4n sized blocks, the search window is 8+15 pixels high and 4n+15 pixels wide. It is divided in 6 sub-windows as shown in Figure 5. They all have the same height which is the reference block height : 8. As the search window height is not a multiple of 8, sub-windows SWC1 and SWC2 extend beyond the search window : their bottom row is outside the search window. These pixels need not be entered in the circuit, but for timing reasons (loading the sub-window SWC1 must last as long as loading sub-window SWB1 or SWA1), some value must be entered in their place. The leftmost sub-windows SWA1, SWB1 and SWC1, are all 15 pixels wide and are loaded during the initialization sequence. The rightmost sub-windows SWA2, SWB2 and SWC2, are all 4n pixels wide and are loaded during the block sequence.

4.2.1 Initialization Sequence

For an illustration of this section refer to Figure 6.

The initialization sequence for 8x4n sized reference blocks starts with a BSYNC strobe during the first cycle. During the 8 first cycles, no data is entered in the circuit. During this time, the circuit initialises itself. Then loading of blocks SWA1, SWB1 and SWC1 starts simultaneously through ports A[7:0], B[7:0], C[7:0], respectively. When this loading ends, the sequence is terminated. Its duration is 128 cycles.

Figure 5 : Search Window for 8 x 4n Reference Block

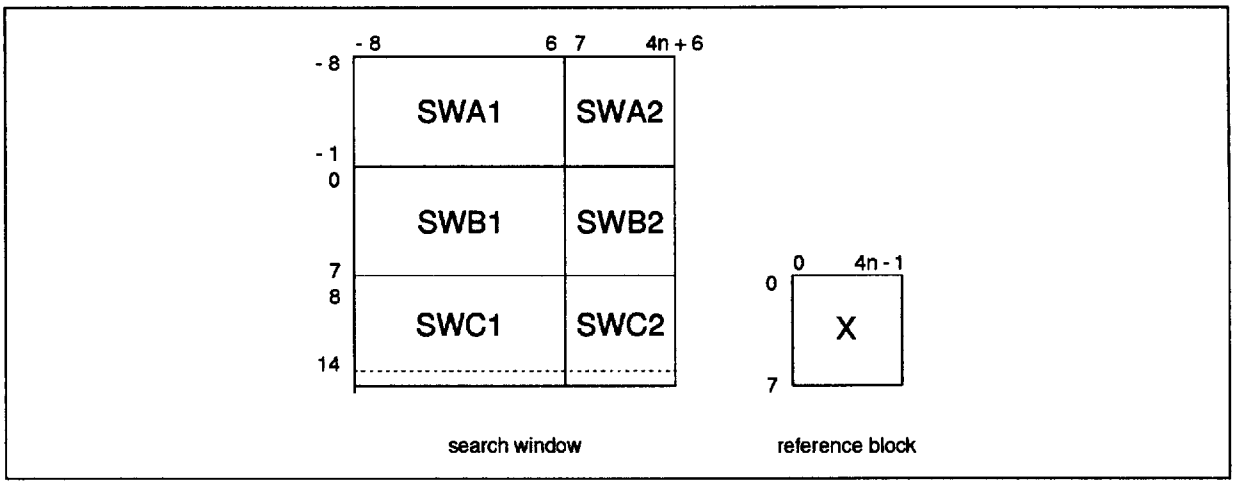
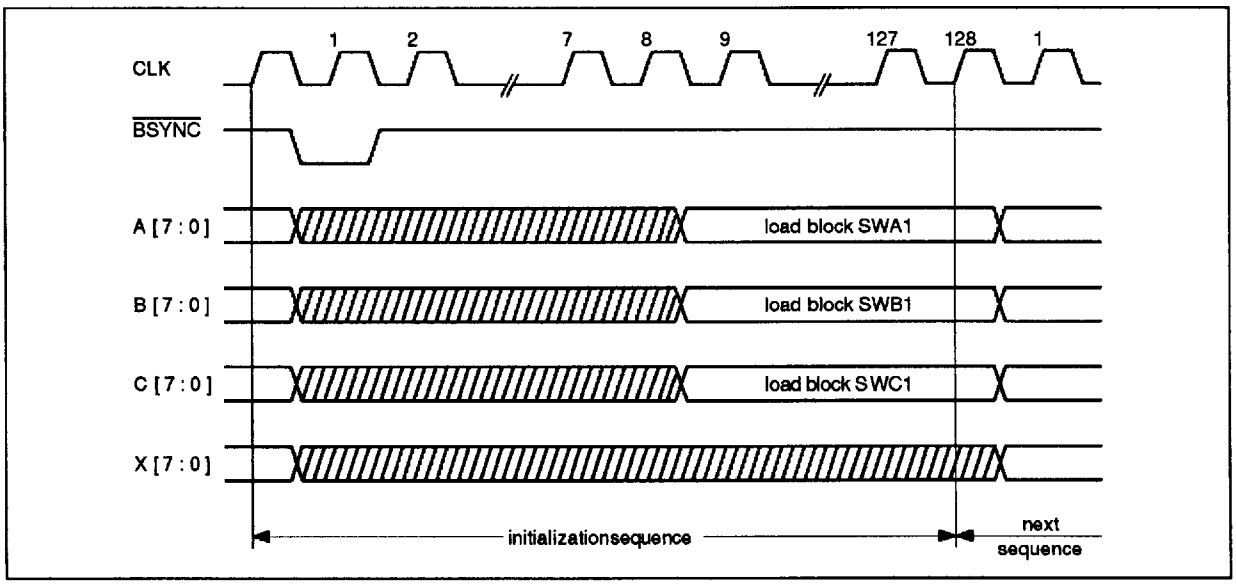


Figure 6 : Initialization Sequence for 8 x 4n Blocks



4.2.2 Block Sequence

For an illustration of this section refer to Figure 7.

When processing rectangular reference blocks (SQ high), a BSYNC low strobe during the first cycle of every block sequence is compulsory. When processing square reference blocks (SQ low), this strobe is optional. During a block sequence, the reference block and sub-windows SWA2, SWB2 and SWC2 are loaded simultaneously through ports X [7 : 0], A [7 : 0], B [7 : 0], C [7 : 0], respectively. This sequence takes exactly 8 x 4n = 32n cycles as all these blocks have the same size.

4.3 16x4n blocks

For 16x4n sized blocks, the search window is 16+15 pixels high and 4n+15 pixels wide. It is divided in 6 sub-windows as shown in Figure 8. They all have the same height which is the reference block height : 16. Sub-windows SWA1, SWA2, SWC1 and SWC2 extend largely beyond the search window. These pixels need not be entered in the circuit, but for timing reasons (loading the sub-windows SWA1 and SWC1 must last as long as loading sub-window SWB1), some value must be entered in their place. An alternative method may be used to enter the required data items without entering unused values. See section 4.3.1.

The leftmost sub-windows SWA1, SWB1 and SWC1, are all 15 pixels wide and are loaded during the initialization sequence. The rightmost sub-windows SWA2, SWB2 and SWC2, are all 4n pixels wide and are loaded during the block sequence.

Figure 7 : Block Sequence for 8 x 4n Blocks

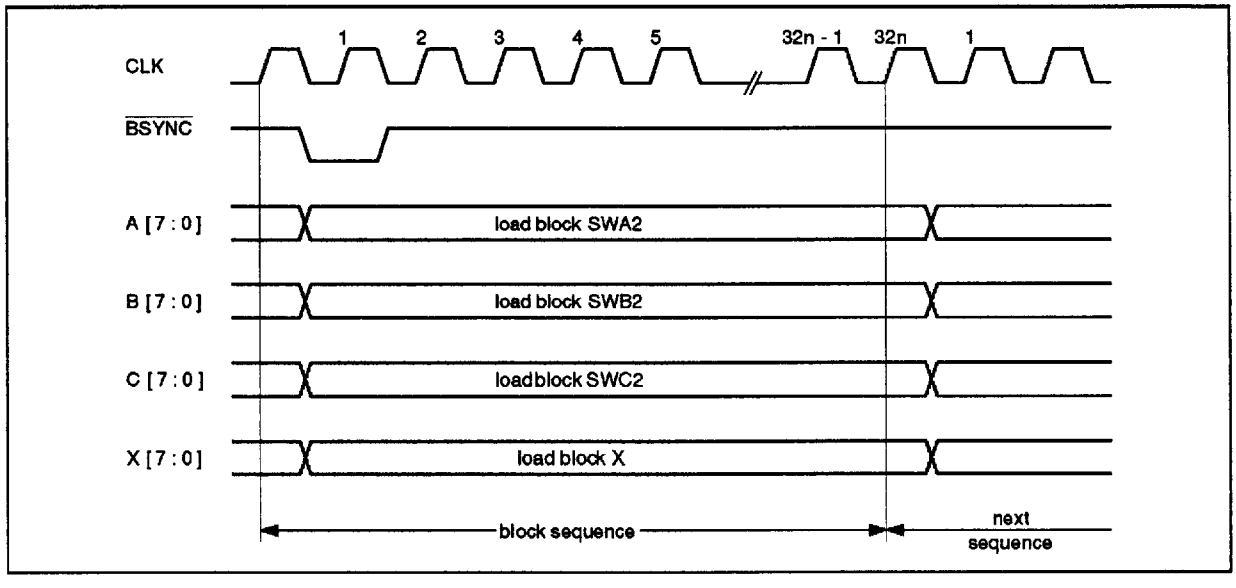
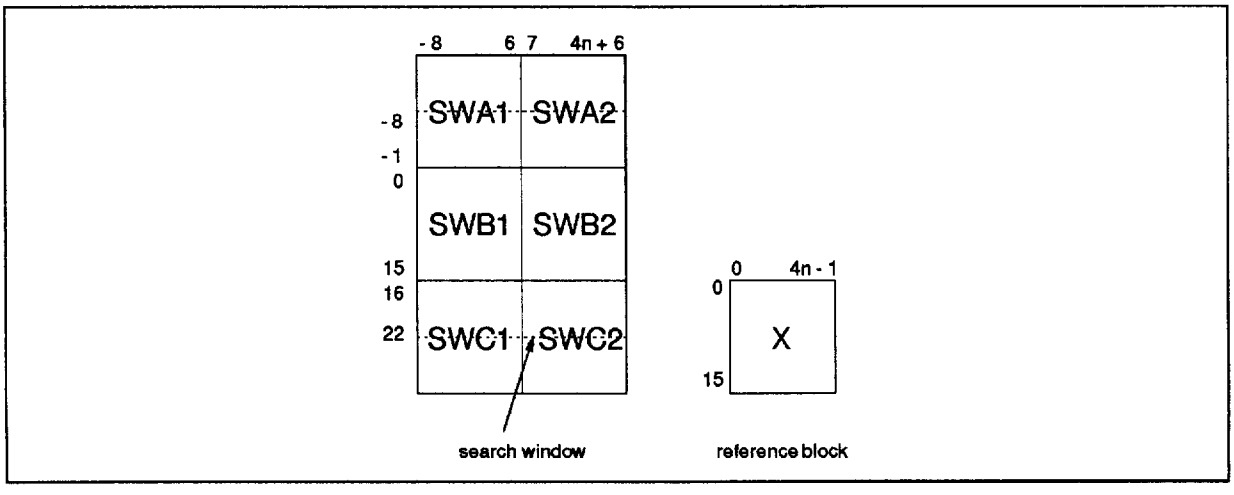


Figure 8 : Search Window for 16 x 4n Reference Blocks



4.3.1 Initialization Sequence

For an illustration of this section refer to Figure 9.

The initialization sequence for 16x4n sized reference blocks starts with a $\overline{\text{BSYNC}}$ strobe during the first cycle. During the 16 first cycles, no data is entered in the circuit. During this time, the circuit initializes itself. Then loading of blocks SWA1, SWB1 and SWC1 starts simultaneously through ports A [7 : 0], B [7 : 0], C [7 : 0], respectively. When this loading ends, the sequence is terminated. Its duration is 256 cycles.

It is worth noting that the useful pixels of block SWC1 (those inside the search window) and the unused pixels of block SWA1 (those outside the search window) are entered in the circuit concurrently, and vice versa. This is because these two blocks have symmetric positions in relation to the search window. Thus, the unused pixels of block SWA1 can be the useful pixels of block SWC1, and vice versa. Practically, this is realised by linking ports A [7 : 0] and C [7 : 0] and writing to this common port a block composed of the 8 upper rows of block SWC1 and the 8 lower rows of block SWA1 (see section 5.3 for a system diagram). However, this composed block still contains unused data : the 8th row of block SWC1, which is outside the search window.

4.3.2 Block Sequence

For an illustration of this section refer to Figure 10.

When processing rectangular reference blocks ($\overline{\text{SQ}}$ high), a $\overline{\text{BSYNC}}$ low strobe during the first cycle of every block sequence is compulsory. When processing square reference blocks (SQ low), this strobe is optional.

During a block sequence, the reference block and sub-windows SWA2, SWB2 and SWC2 are loaded simultaneously through ports X[7:0], A[7:0], B[7:0], C[7:0], respectively. This sequence takes exactly 16x4n=64n cycles as all these blocks have the same size.

The ports A[7:0] and C[7:0] can be linked and the blocks SWC2 and SWA2 partially scanned as seen above to reduce the number of data items delivered to the circuit.

Figure 9 : Initialization Sequence for 16 x 4n Blocks

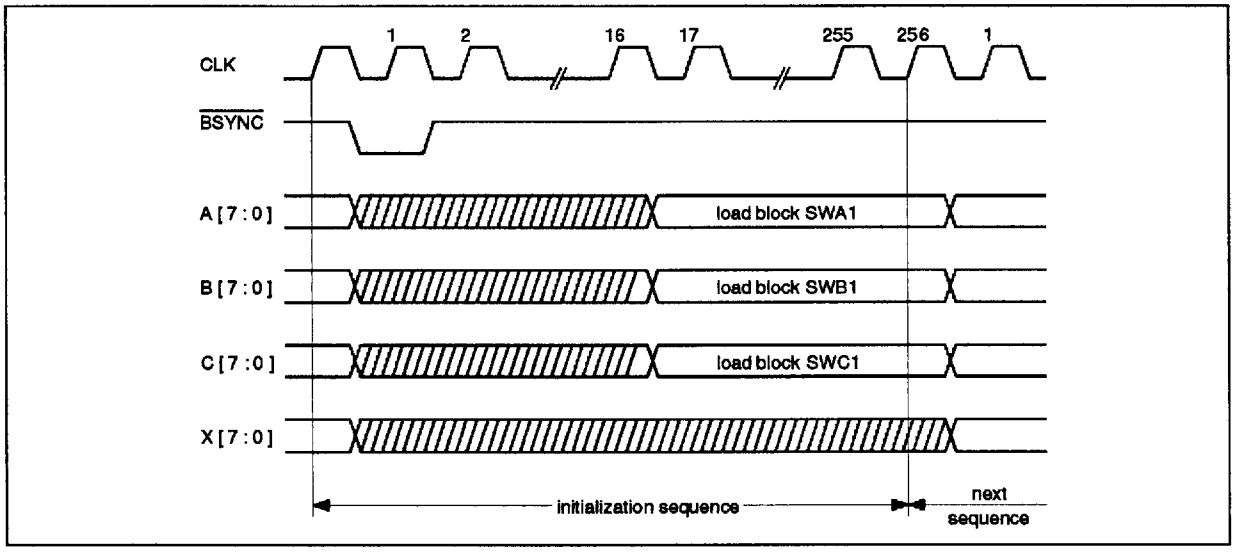
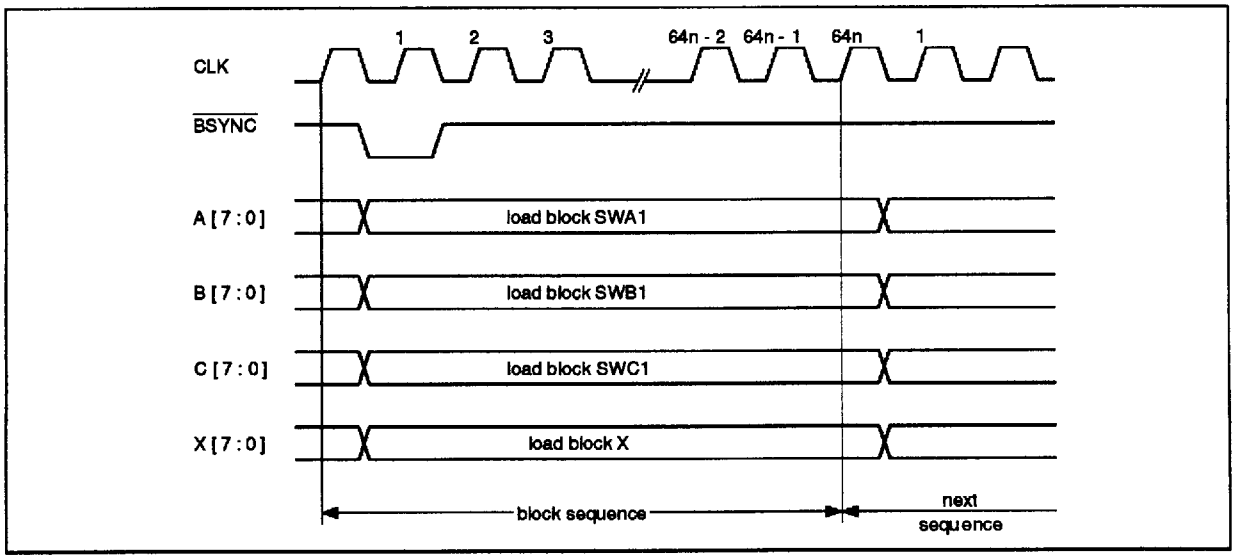


Figure 10 : Block Sequence for 16 x 4n Blocks



4.4 Actual Operation

Actual operation of the circuit is a succession of initialization sequences and block sequences. No dead cycle can be introduced between the successive sequences, as otherwise the circuit pipeline is broken.

4.4.1 Processing One Block

Processing one block requires an initialization sequence immediately followed by a block sequence. During the initialization sequence, the 15 left columns of the search window are entered and stored in the circuit and no computation is performed. During the block sequence, the reference block and the remaining part of the search window are entered and processed in the circuit. During this sequence, the 256 distortions are computed. The output of the minimum distortion and the motion vector takes place a few cycles after the end of the block sequence, see section 4.6 for more details.

4.4.2 Continuous Processing of Blocks

One way to process several blocks is to repeat the above described operations as many times as necessary. However, under certain conditions, there is a more efficient way of processing several blocks. Under these conditions, only one initialization sequence and as many block sequences as there are reference blocks are necessary. The gain is substantial : there is a factor 3 (8x4n blocks) or 2 (16x4n blocks) gain in memory bandwidth and operating frequency.

The only condition to be observed in using this mode is that the 15 rightmost columns of search window k are exactly the 15 leftmost columns of search window k+1. The reason why this mode is more efficient is easily understandable. The circuit takes advantage of the fact that successive search windows overlap. At the end of block sequence k, the 15 leftmost columns of search window k+1 are already loaded in the circuit so that no initialization sequence is required for block k+1.

The first and unique initialization sequence loads in the 15 leftmost columns of the first search window. It is immediately followed by the successive block sequences. During block sequence k, results relative to block sequence k-1 are accessible through the I/O port.

In particular, continuous processing of blocks is possible when reference block k is the right neighbour of reference block k-1. In this case the condition that two successive search windows overlap by 15 columns is met.

If the edge control is set correctly, the pipeline is not broken when a search window is situated near the right edge of a frame and does not therefore overlap with the next search window. For more details on how to do this, see section 5.

4.5 The Block Synchronisation signal : $\overline{\text{BSYNC}}$

An initialisation sequence always starts with a $\overline{\text{BSYNC}}$ strobe (active low). This sets the circuit ready for operation.

When rectangular blocks are processed ($\overline{\text{SQ}}$ high), the circuit must be informed when a new block starts. This is done by strobing $\overline{\text{BSYNC}}$ low during the first cycle of every block sequence.

When square blocks are processed ($\overline{\text{SQ}}$ low), the circuit knows the size of the block so that the $\overline{\text{BSYNC}}$ strobe is optional during block sequences.

Signal $\overline{\text{H8}}$ and $\overline{\text{SQ}}$ are read in the circuit at each $\overline{\text{BSYNC}}$ strobe.

4.6 Output of Results

Refer to Figure 11.

At the beginning of every sequence, the circuit computes the minimum and its address among the values stored in the distortion RAM. If the previous sequence was a block sequence, the circuit computes the motion vector and the minimum distortion relative to the reference block entered during this sequence. If it was an initialization sequence, the results are meaningless.

When the results are available, the $\overline{\text{MVSYNC}}$ signal goes low. This happens during the (M+30)th cycle of every sequence (M=8 or 16). What happens next depends on the state of the IOB bus.

If OE is high, the motion vector is present on the IOB bus during one cycle, while $\overline{\text{MVSYNC}}$ is low. Then $\overline{\text{MVSYNC}}$ goes high on the next cycle, during which nothing happens. Then $\overline{\text{DSYNC0}}$ goes low while the minimum distortion msb is present on IOB. Then $\overline{\text{DSYNC0}}$ goes high again and $\overline{\text{DSYNC1}}$ goes low while the minimum distortion lsb is present on IOB. If OE goes low while the results are outputted, the data not yet delivered is lost.

If OE is low, IOB is in the high impedance state and $\overline{\text{MVSYNC}}$ remains low. When OE goes high again the data is outputted as described above. This feature is useful when several STI3220 circuits are used in parallel : the circuits can share a common bus and although all the results are ready at the same time, the system can read them sequentially. If OE does not go high before a new sequence starts, then $\overline{\text{MVSYNC}}$ returns to a high state and the data is lost.

If, in a given distortion set, several distortions are equal to the minimum of the set, the circuit will output the motion vector which has the highest priority according to Table 2.

There is no simple algorithm which can describe this table, it is just a consequence of the circuit architecture. Note, however, that the (0,0) motion vector has the highest priority.

Table 3 gives the exact bit allocation of bus IOB when results are delivered. Vj[3:0] is the horizontal component of the motion vector and is coded in 2s complement. Similarly, Vi[3:0] is the vertical component of the motion vector. The minimum distortion is an unsigned value.

Table 2

	-8	-7	-6	-5	-4	-3	-2	-1	0	1	2	3	4	5	6	7
-8	2	12	30	58	90	122	154	186	187	155	123	91	59	31	13	3
-7	0	6	20	42	74	106	138	170	171	139	107	75	43	21	7	1
-6	8	22	44	76	108	140	172	202	203	173	141	109	77	45	23	9
-5	4	14	32	60	92	124	156	188	189	157	125	93	61	33	15	5
-4	18	40	72	104	136	168	200	226	227	201	169	137	105	73	41	19
-3	10	28	56	88	120	152	184	214	215	185	153	121	89	57	29	11
-2	26	54	86	118	150	182	212	236	237	213	183	151	119	87	55	27
-1	16	38	70	102	134	166	198	224	225	199	167	135	103	71	39	17
0	36	68	100	132	164	196	222	244	255	223	197	165	133	101	69	37
1	24	52	84	116	148	180	210	234	235	211	181	149	117	85	53	25
2	50	82	114	146	178	208	232	249	250	233	209	179	147	115	83	51
3	34	66	98	130	162	194	220	242	243	221	195	163	131	99	67	35
4	64	96	128	160	192	218	240	253	254	241	219	193	161	129	97	65
5	48	80	112	144	176	206	230	247	248	231	207	177	145	113	81	49
6	62	94	126	158	190	216	238	251	252	239	217	191	159	127	95	63
7	46	78	110	142	174	204	228	245	246	229	205	175	143	111	79	47

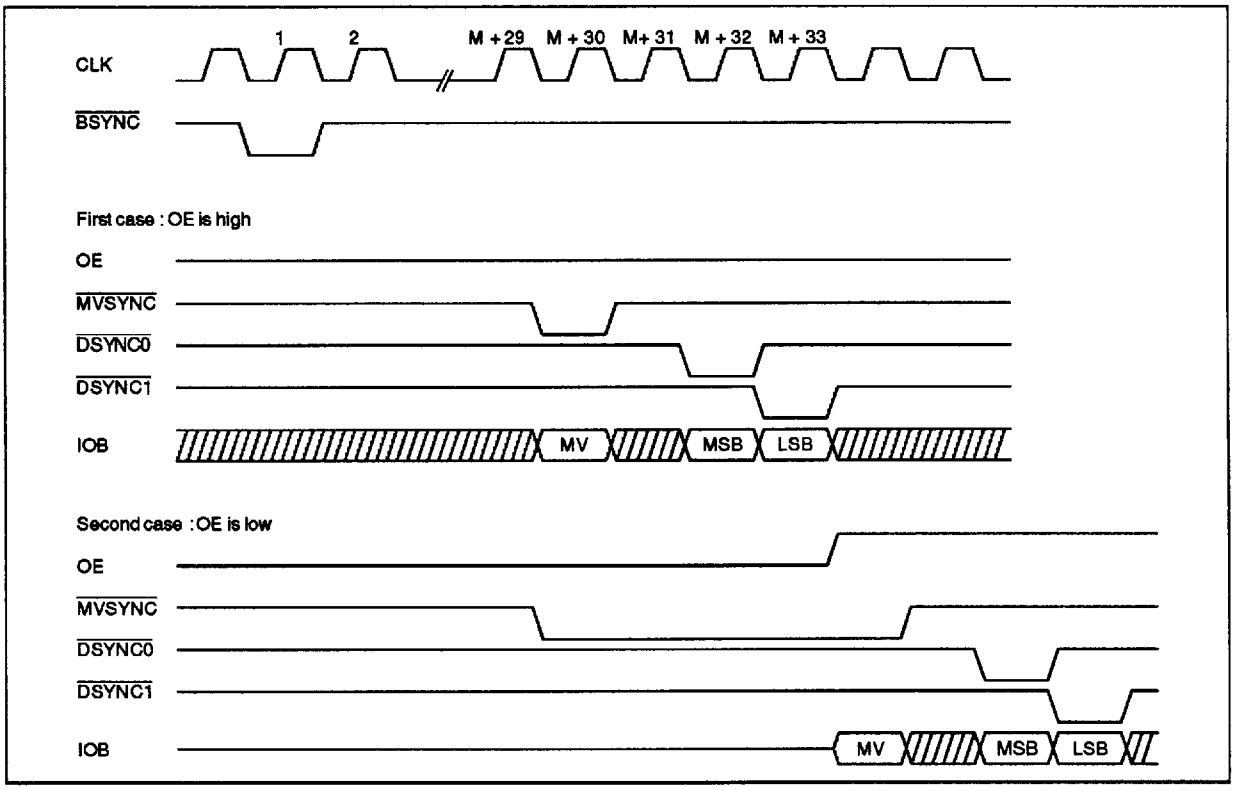
3220-03.TBL

Table 3

	IOB7	IOB6	IOB5	IOB4	IOB3	IOB2	IOB1	IOB0
Motion Vector	Vj3	Vj2	Vj1	Vj0	Vi3	Vi2	Vi1	Vi0
Distortion MSB	D15	D14	D13	D12	D11	D10	D9	D8
Distortion LSB	D7	D6	D5	D4	D3	D2	D1	D0

3220-04.TBL

Figure 11 : Output of Results



3220-12.EPS

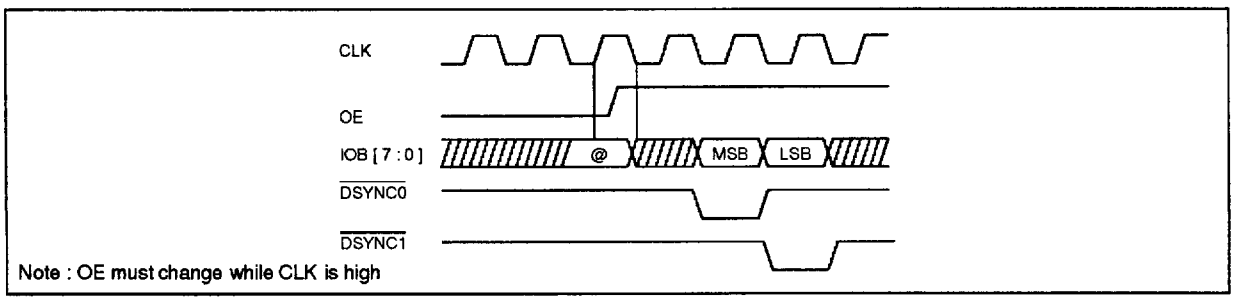
4.7 How to Read a Distortion

When a minimum computation is not in process, the distortion RAM can be read by the system through the IOB bus. This particular mode of operation is illustrated in Figure 12.

The address must be entered while the bus is in high impedance state (OE low). Then OE must go high in order to let the circuit write on the bus, (OE signal must change value while CLK is high). The address taken into account is the one present on the bus during the last rising edge of CLK when OE was low. One dead cycle elapses, and the distortion is written out, first the msbs then the lsbs. Four cycles are thus necessary to read a distortion.

The address of a distortion is the related motion vector. The bit allocation is exactly the same as seen above in section 4.6.

Figure 12 : Reading a Distortion



3220-13.EPS

When a minimum computation is in process, the distortion RAM cannot be read.

M + 1 cycles after the beginning of every sequence, a new set of distortions is stored in the RAM. These distortions are those related to the previous sequence. A minimum computation immediately follows and lasts 28 cycles as shown in Figure 13.

Then the results are delivered and this takes another 4 cycles. The RAM is then available and distortions of the new set can be read until another set is stored in RAM, M + 1 cycles after the beginning of the next sequence.

4.8 How to Use the Edge Control

When a search window is partially outside the actual frame, the number of candidate blocks is less than 256 and the set of valid motion vectors is reduced. The STi3220 can take this into account and compute motion vectors in the valid range. There are 9 different cases depending on whether the search window is along one of the edges or in one of the corners of a frame.

To set the edge control, four pins are provided. They are all latched in the circuit during the first cycle of every block sequence and are related to the search window loaded during this very sequence.

Each pin is related to one edge of the frame (left, right, top or bottom) and setting one pin high during the first cycle of a block sequence will mean that the current search window (the one being loaded into the circuit during this particular block sequence) extends beyond the edge of the frame related to that pin, and that the valid set of motion vectors for the current search window is reduced.

The most common case is when the search window is entirely included in the frame, and the four pins are set low.

Pin EC0 is related to the top edge of the frame, and setting this pin high will reduce the valid range of motion vectors to the ones with a positive or null vertical component.

Pin EC1 is related to the right edge of the frame, and setting this pin high will reduce the valid range of motion vectors to the ones with a negative or null horizontal component.

Pin EC2 is related to the bottom edge of the frame, and setting this pin high will reduce the valid range of motion vectors to the ones with a negative or null vertical component.

Pin EC3 is related to the left edge of the frame, and setting this pin high will reduce the valid range of motion vectors to the ones with a positive or null horizontal component.

When the search window extends beyond a corner of the frame, two edges have to be selected.

Unrealistic combinations of edges are not prohibited and will yield logical results. For example, selecting the top **and** the bottom edge will reduce the valid set of motion vectors to the horizontal ones. All combinations are summarized in the Table 4. Unrealistic combinations are indicated with a*.

EC_n = 0 unless otherwise noted.

Figure 13 : RAM Availability

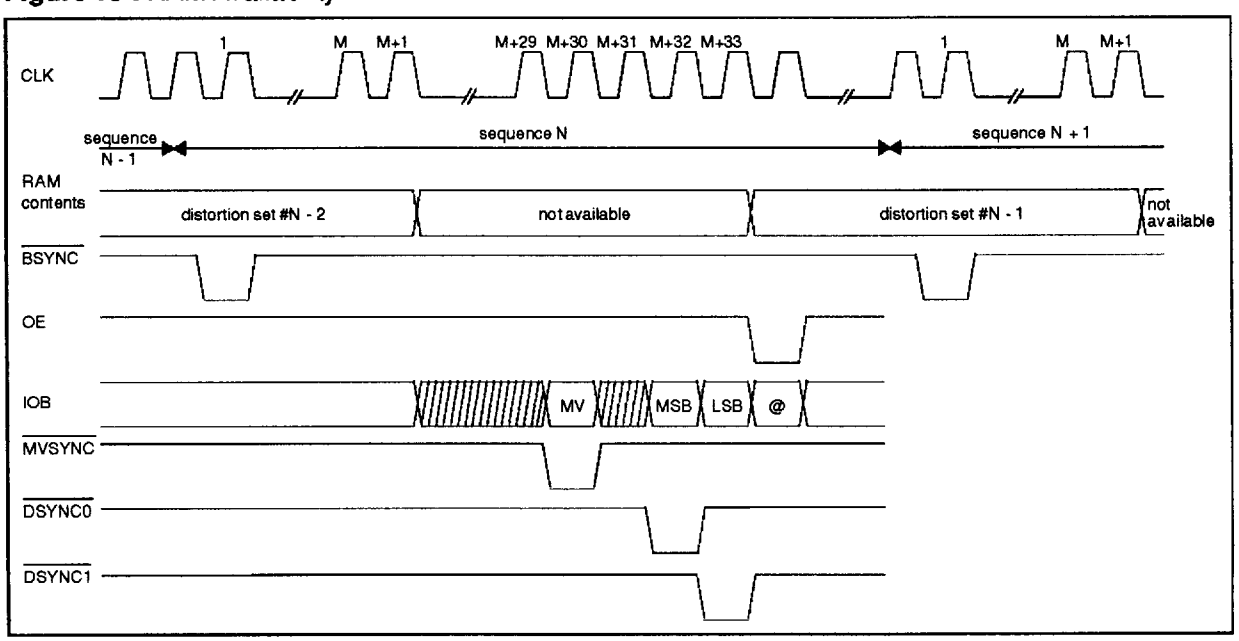
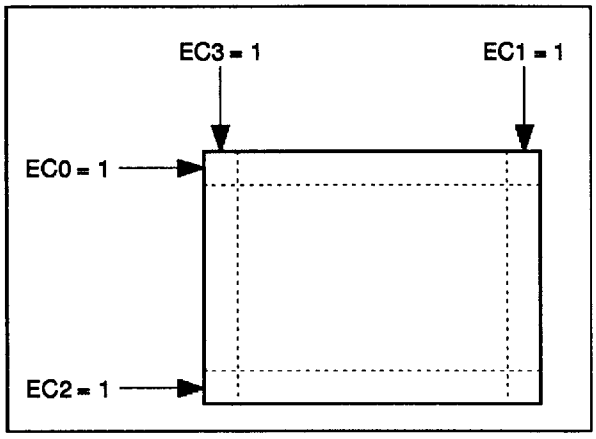


Table 4

				Valid Range of Motion Vectors	
EC3	EC2	EC1	EC0	Horizontal Component	Vertical Component
0	0	0	0	$-8 \le j \le 7$	$-8 \le i \le 7$
0	0	0	1	$-8 \le j \le 7$	$0 \le i \le 7$
0	0	1	0	$-8 \le j \le 0$	$-8 \le i \le 7$
0	0	1	1	$-8 \le j \le 0$	$0 \le i \le 7$
0	1	0	0	$-8 \le j \le 7$	$-8 \le i \le 0$
0*	1	0	1	$-8 \le j \le 7$	$i = 0$
0	1	1	0	$-8 \le j \le 0$	$-8 \le i \le 0$
0*	1	1	1	$-8 \le j \le 0$	$i = 0$
1	0	0	0	$0 \le j \le 7$	$-8 \le i \le 7$
1	0	0	1	$0 \le j \le 7$	$0 \le i \le 7$
1*	0	1	0	$j = 0$	$-8 \le i \le 7$
1*	0	1	1	$j = 0$	$0 \le i \le 7$
1	1	0	0	$0 \le j \le 7$	$-8 \le i \le 0$
1*	1	0	1	$0 \le j \le 7$	$i = 0$
1*	1	1	0	$j = 0$	$-8 \le i \le 0$

3220-05.TBL

Figure 14



3220-15.EPS

4.9 How to Set the Block Size : signals $\overline{SQ}$ and $\overline{H8}$

To set the block size, two pins are provided. Both are latched in when $\overline{BSYNC}$ is strobed low.

To process 8 pixel high reference blocks, signal $\overline{H8}$ must be low during the $\overline{BSYNC}$ strobe. To process 16 pixel high reference blocks, signal $\overline{H8}$ must be high during the $\overline{BSYNC}$ strobe.

To process square reference blocks, signal $\overline{SQ}$ must be low during the $\overline{BSYNC}$ strobe. To process rectangular reference blocks, signal $\overline{SQ}$ must be high during the $\overline{BSYNC}$ strobe.

Table 5 summarizes the four different block sizes.

Table 5

$\overline{H8}$	$\overline{SQ}$	Block Size
0	0	8 x 8
0	1	8 x 4n
1	0	16 x 16
1	1	16 x 4n

When the circuit is set to process rectangular blocks, it does not know the width of the block and so must be told when a new block starts. This is why, when rectangular blocks are processed, a $\overline{BSYNC}$ strobe is compulsory during the first cycle of every block sequence. It may be noted, for reference purposes only, that in this mode the successive reference blocks need not have the same width, provided their width is a multiple of 4. When the circuit is set to process square blocks, $\overline{BSYNC}$ strobes are optional at the beginning of every block sequence. They are required only with initialization sequences. If the circuit is used in pipeline mode as explained in section 4.4.2, then only one $\overline{BSYNC}$ strobe is needed during the unique initialization sequence.

4.10 How to Reset the circuit : signal $\overline{RESET}$

The STi3220 must be reset before operation. This is done by strobing the signal $\overline{RESET}$ low during at least one cycle before any initialization sequence starts. The clock must be running.

4.11 How to measure the circuit latency time : signal $\overline{OSYNC}$

The circuit latency time Tlat is the time elapsing between the input of the first pixel of a reference block, and the output of the motion vector relative to this block.
The motion vector is ready a few cycles after the last pixel of the reference block is entered in the circuit, 38 cycles for 8x4n sized reference blocks and 46 cycles for 16x4n sized reference blocks. The latency time is thus 32n+38 cycles for 8 x 4n sized reference blocks and 64n+46 cycles for 16 x 4n sized reference blocks.
To ease system implementation when this latency time must be taken into account, e.g. for the initialization of circuits expecting results from the STi3220, the $\overline{OSYNC}$ signal is provided.
The circuit generates one $\overline{OSYNC}$ low strobe for every $\overline{BSYNC}$ strobe. This occurs exactly Tlat cycles after the related $\overline{BSYNC}$ strobe, concurrently with the MVSYNC strobe indicating that the motion vector is ready.

4.12 How to freeze the circuit : signal $\overline{EN}$

It may be useful to freeze the circuit in an asynchronous way for an indeterminate period of time without breaking the circuit pipeline or loosing any data. This is possible with the STi3220 because it is a fully static CMOS device.
When signal $\overline{EN}$ is high, the circuit internal clock is frozen until signal $\overline{EN}$ goes low again.
To ensure no loss of data during the transitions between the frozen and the working states of the circuit, the $\overline{EN}$ signal must fulfill the requirements specified in section 6.

5 - BASIC STi3220 APPLICATION EXAMPLES

5.1. - Picture borders : example with 8*8 blocks, displacement range -8 to +7:

When computing a block in the middle of a picture in pipeline mode, the left and middle parts of the search window (Le and Mi columns) are already loaded into the STi3220 and the right part (Ri columns) is input with the reference block. The position of that right area in the picture is shifted 8 columns right from the reference block position (see Figure 15).

However when the reference block is the last one of a row of blocks in the image, only one row for the search window is available on the right of this reference block. In order not to break the pipeline mode, the remaining 7 columns of the search window will be the 7 columns corresponding to the next block that will be evaluated in the pipeline i.e. the first block position of the next row of block (see Figure 16). Those 7 columns are loaded into the chip but will not be taken into account for the motion vector calculation if the Edge Control signals are set to EC3...EC0 = 0010 (right hand edge).

The first reference block of the next line is entered in the chip in synchronism with search window columns from 7 to 15 (see Figure 17). The middle part of the search window has already been loaded during the previous block and the left part of the search window is not taken into account for motion vector calculation by setting the Edge Control bits to EC3...EC0 = 1000 (left hand edge).

Of course the same principle can be applied when reaching the end of an image (see Figure 18): the search window corresponding to the first block position of an image is loaded during the processing of the last block of an image : pipeline processing is never broken. Just notice in that case that the first column inputted is only significant on the A and B bands (last column of the image) and that the 7 following columns are only significant on B and C bands (first columns of the image).

The pipeline continuity on the borders of an image is always respected whatever the blocks' size is.

Figure 15

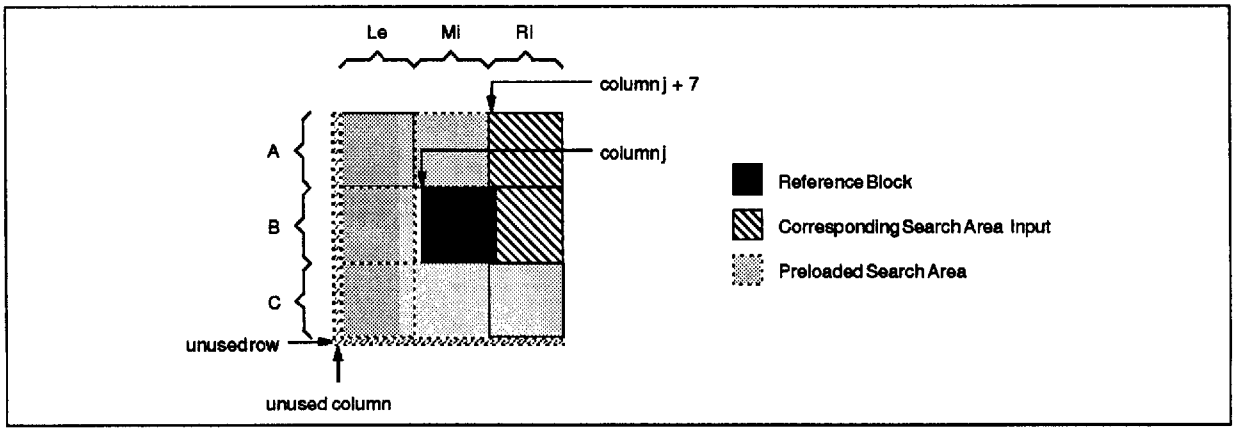


Figure 16

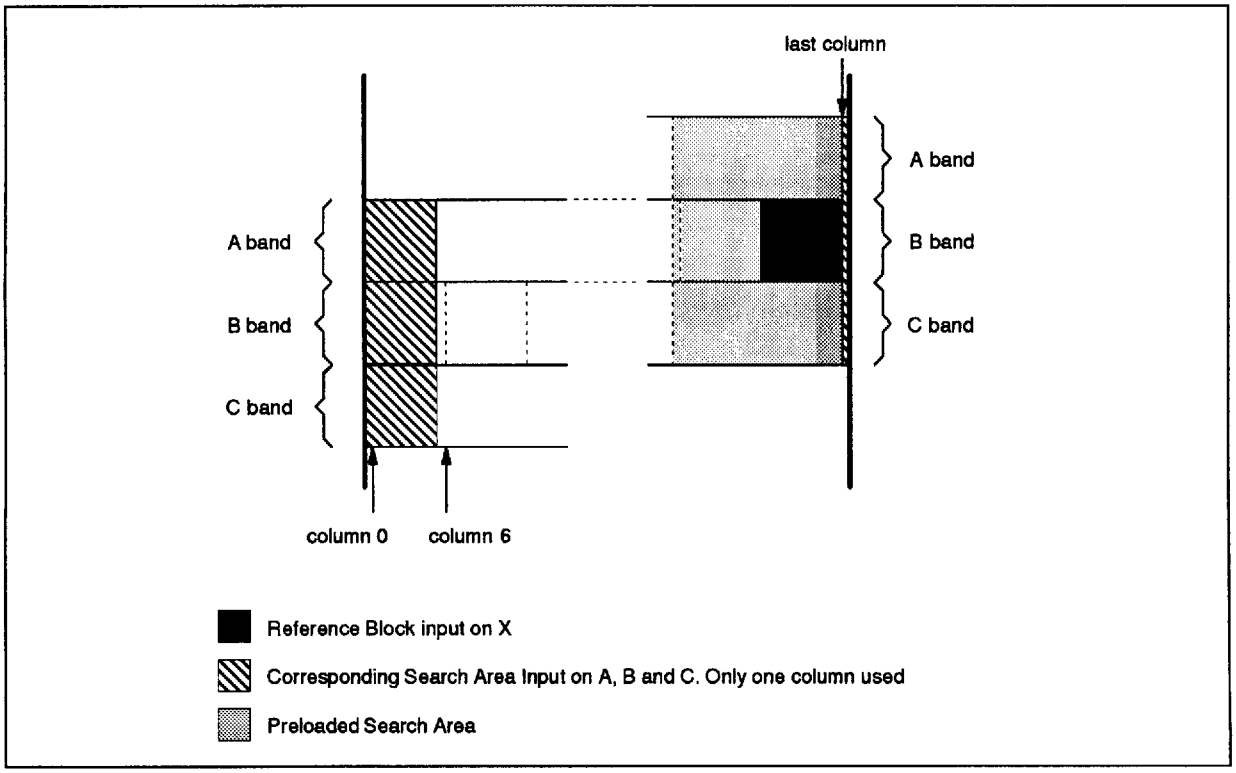


Figure 17

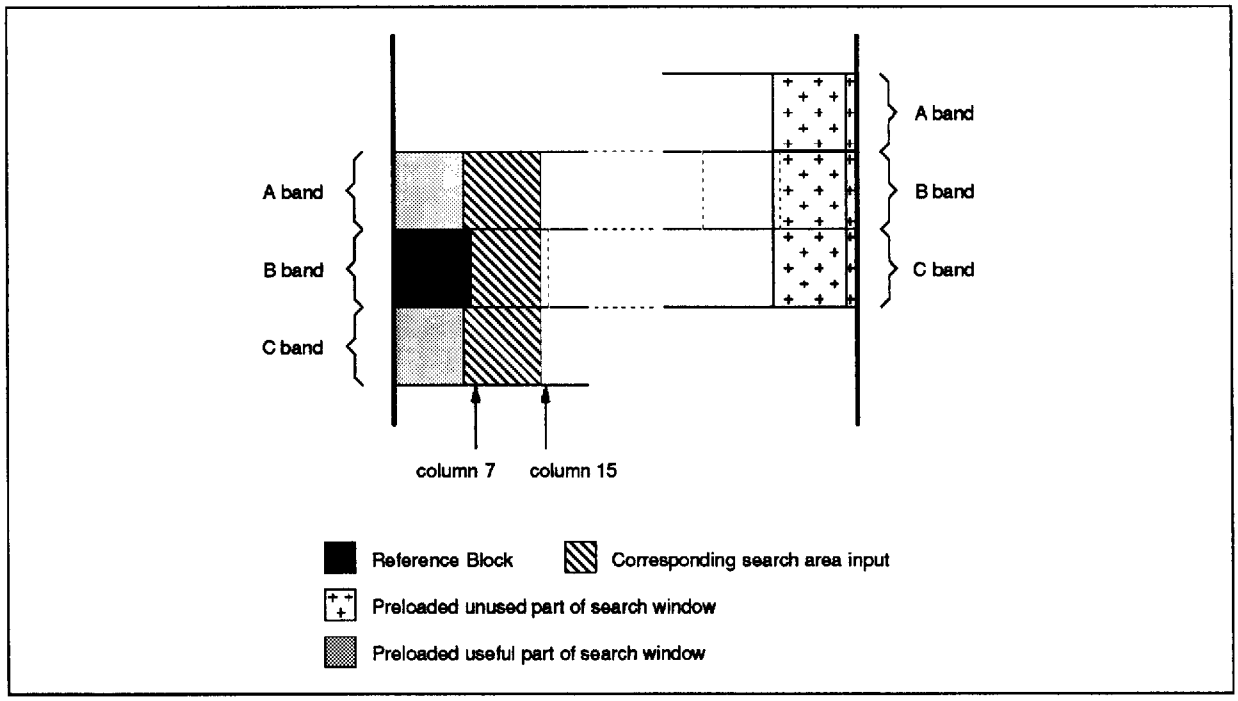
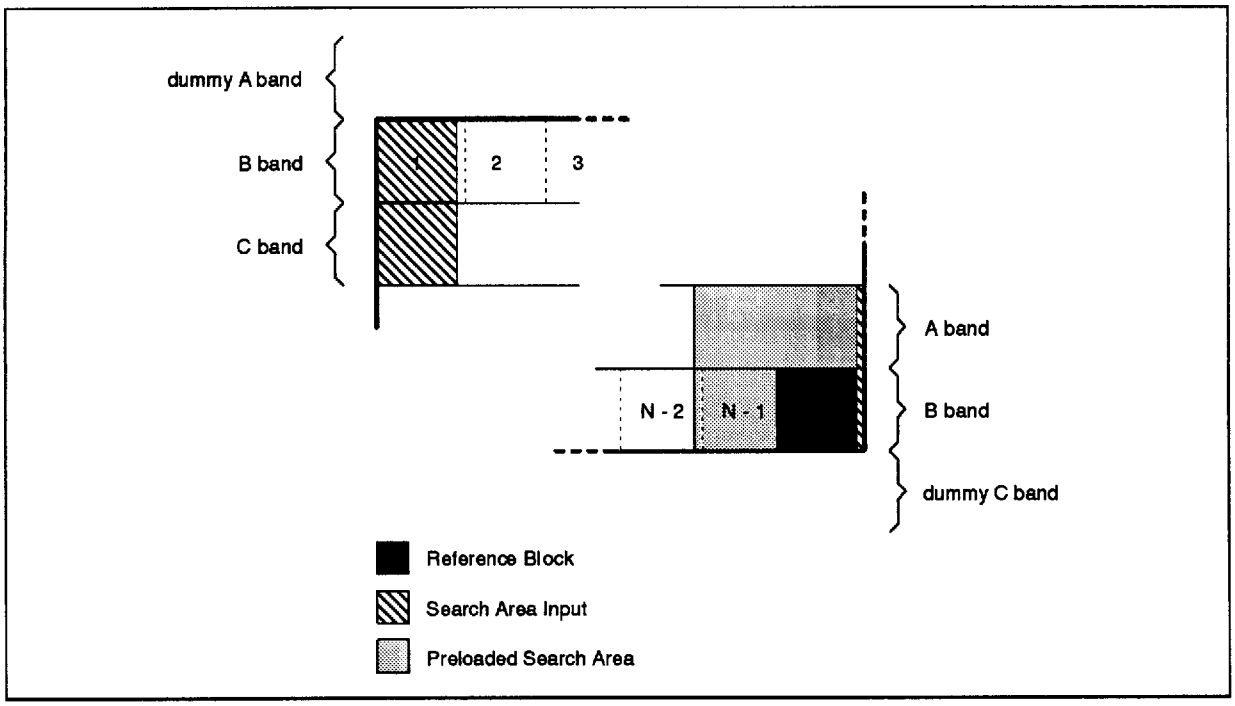


Figure 18



5.2. - Example of system outline in 8 x 4n mode

use of delay lines: in synchronism with the reference block, the frame memory is accessed on the line of blocks under the reference one. Two delay lines are necessary to provide the STi3220 with the middle and upper band of the search window.

For low rate applications it may be cheaper to suppress the delay lines and to access 3 consecutive times into the frame memory for each pixel of the reference block (see Figure 20).

Figure 19

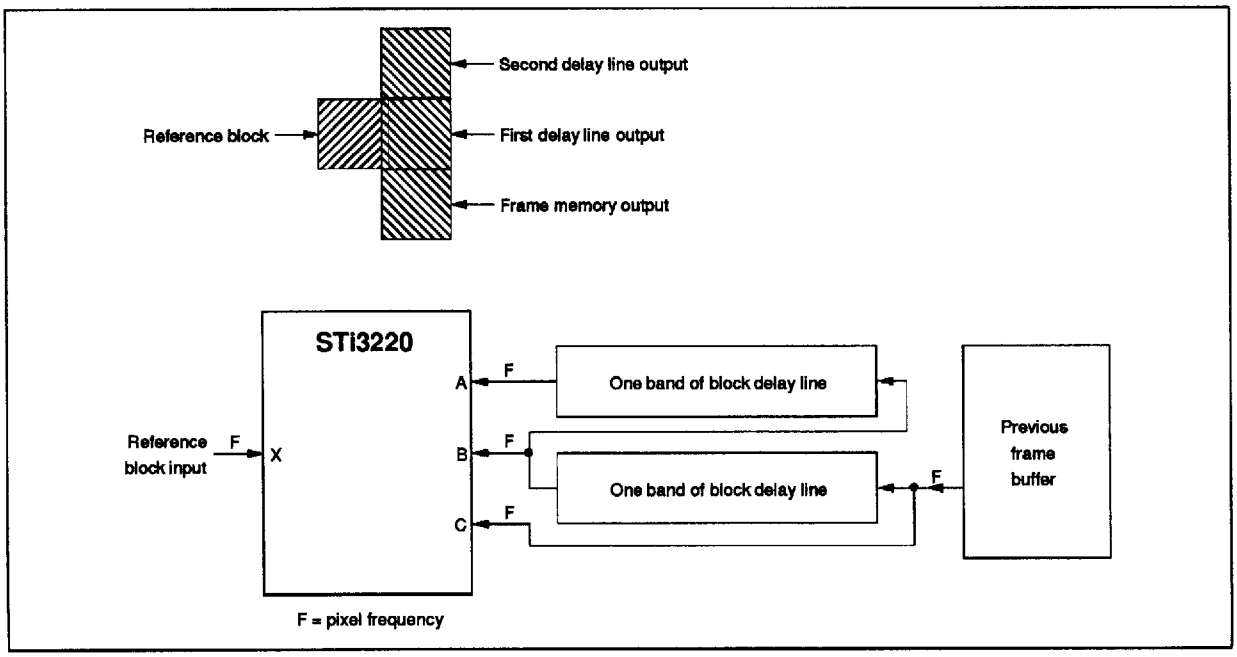
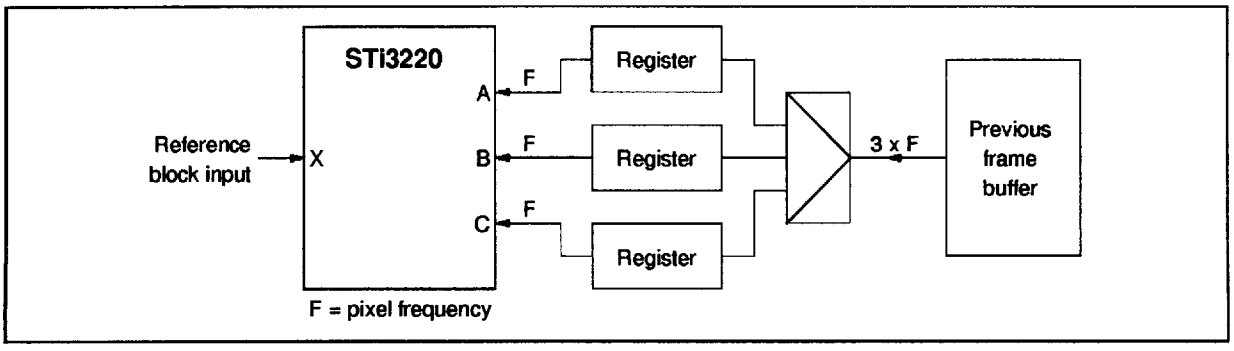


Figure 20



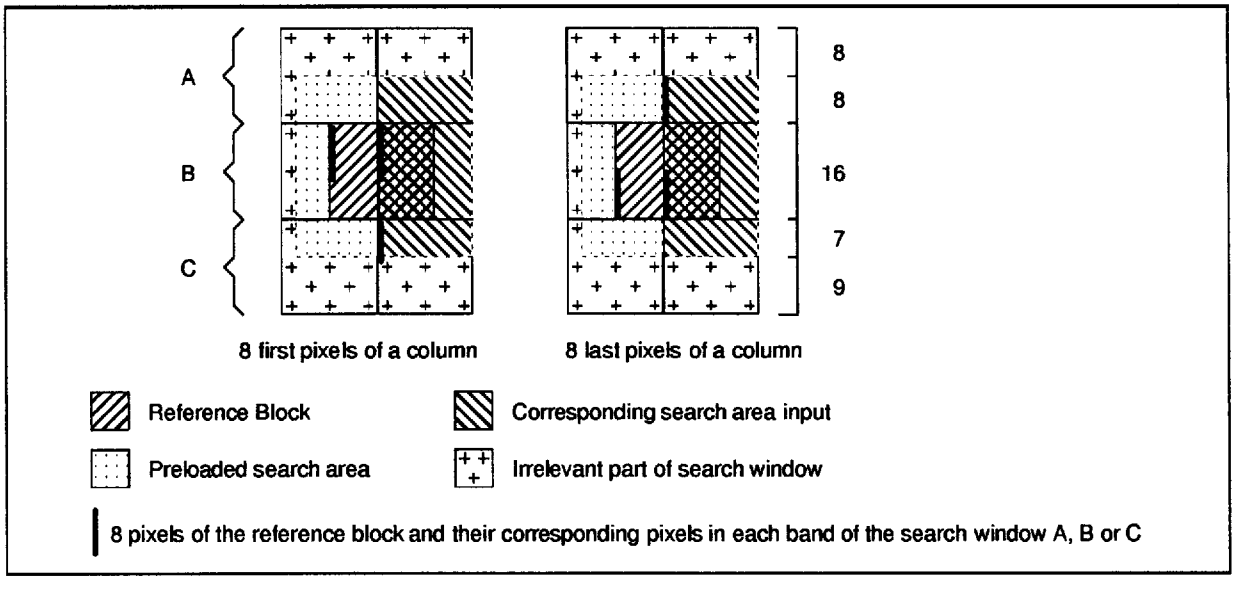
3220-21.EPS

5.3. - Example of system outline in 16*16 mode

In this case the search window is only one half block high above the reference block (8-pixels high) and one half block high under (in fact 7-pixels high). The total height of the search window is only two blocks high. When inputting the 8 first pixels of a column of the reference block, only the 8 corresponding pixels of the search window on band B and C are taken into account. In the same way, when inputting the 8 last pixels of a column of the reference block, only the 8 corresponding pixels of the A and B bands are taken into account.

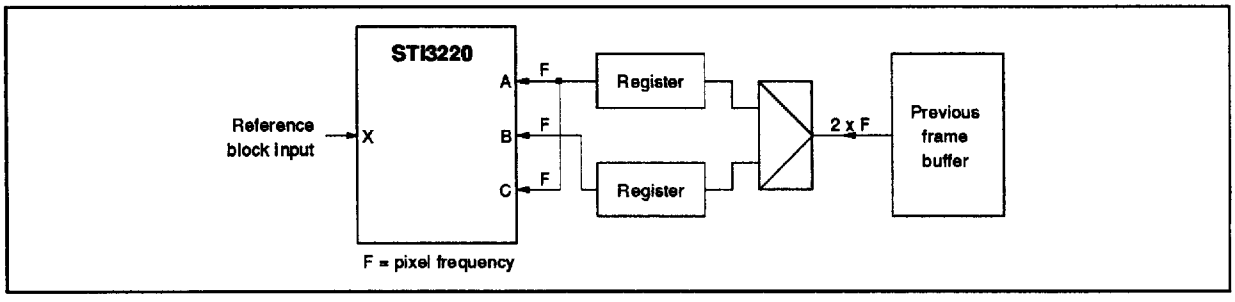
A system architecture using a band by band scanning and 3 delay lines may be implemented in this case (refer to 8*4n example of system outline). For low rate applications it is possible to access only twice the pixel rate on the frame memory in order to provide the search window : as bands A and C are not read at the same time by the chip, they can be connected together (see Figure 22).

Figure 21



3220-22.EPS

Figure 22



3220-23.EPS

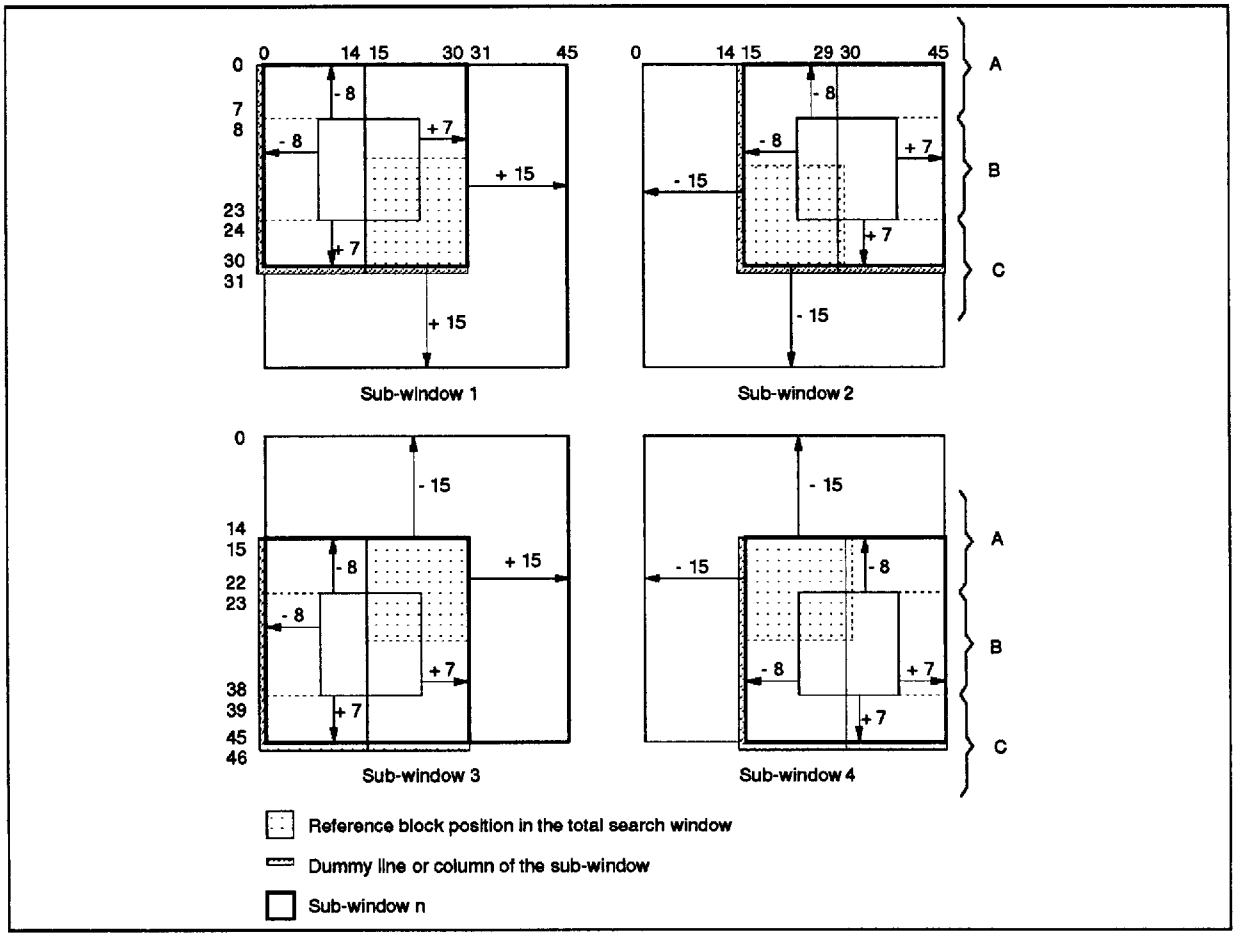
5.4 - principle of +15/-15 research (16*16 blocks)

Computing +15/-15 displacements with the STi3220 chip able to compute +7/-8 displacements is made possible by dividing the total search window into 4 sub search windows (see Figure 23) : 4 corresponding motion vectors and minimum distorsions will be calculated and the system must manage those results in order to decide what will be the final motion vector for all the search window. The four partial motion vectors can be computed in two ways:

- using 4 STi3220 chips each one loaded at the same time with the same reference block and one specific search window per chip. The computation costs 2 blocks (1 initialization sequence + 1 block sequence).
- using only one STi3220 chip loaded 4 consecutive times with the same reference block and each of the specific search window. As the recovery between the different sub search windows is not a multiple of 16, the pipeline mode cannot be used and the computation of the +15/-15 displacement costs 8 blocks 16*16 (4 initialization sequences + 4 block sequences):
  - phase 1 : initialization sequence of sub-window 1 (first 15 columns). Input of a dummy reference block.
  - phase 2 : block sequence of sub-window 1 : input of the 16 last columns of the sub-window 1 and the reference block.
  - phase 3 : initialization sequence of sub-window 2. Result of the first sub-window (motion vector + minimum distortion) obtained after the 46th cycle of that phase.
  - phase 4 : block sequence of sub-window 2.
  - phase 5 : initialization sequence of sub-window 3. Result of sub-window 2 after the 46th cycle.
  - phase 6 : block sequence of sub-window 3.
  - phase 7 : initialization sequence of sub-window 4. Result of sub-window 3 after the 46th cycle.
  - phase 8 : block sequence of sub-window 4.

The result for sub-window 4 is obtained after the next 46 cycles, and hence will typically be available during the beginning of the computations for the next reference block.

Figure 23



It is possible in this way to compute the displacement for a CIF format picture as defined by the CCITT : 352 x 288 pixels x 15 Hz = 1.52 Mpixels/s. Eight accesses to the chip for one input pixel means a chip working frequency equal to 12.16 MHz.
Note: Another alternative method of computing a -/+ 15 motion vector is to carry out the calculations on a -16/+15 search window (i.e. two times -8/+7 in both directions) and to clip the motion vector to -15/+15 range (i.e. if motion vector is equal to -16 it is forced to -15). The advantage of using a -16/+15 search window is that the block sequence of sub-window 1 is identical to the initialization sequence of sub-window 2 (idem for sub-windows 3 and 4). That means that the pipeline mode can be used between sub-windows 1 and 2, and sub-windows 3 and 4. The total computation time of the -16/+15 motion vector then costs 6 blocks (2 initialization sequences + 4 block sequences) instead of 8 blocks.

6. ELECTRICAL CHARACTERISTICS

ABSOLUTE MAXIMUM RATINGS

Symbol	Parameter	Value	Unit
V_{CC}	Supply Voltage	6	V
T_{OPER}	Operating Temperature Range	0, 70	°C
	Voltage on any pin relative to V_{SS}	6	V

3220-06.TBL

DC ELECTRICAL CHARACTERISTICS

Operating conditions : V_{SS} = 0V, T_A = 0 to 70°C, V_{CC} = 5V ± 5%, unless otherwise specified

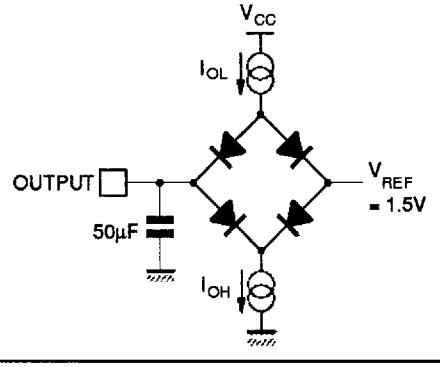
Symbol	Parameter	Conditions	Min.	Typ.	Max.	Unit
V_{CC}	Operating Voltage		4.75		5.25	V
I_{CC}	Supply Current • F_{CLK} = 20MHz • F_{CLK} = 0MHz	C_{LOAD} = 50pF (all outputs) All inputs at V_{CC} or V_{SS}			450 1	mA
V_{IL} V_{IH}	Input Voltage Level • Logic Low • Logic High	V_{CC} = 5 ± 0.25V	 2		0.8	V
	High Impedance Input Leakage • I/O Buffers • Input Buffers	V_{IN} = VSS to VCC	-5 -1		+5 +1	µA
V_{OL} V_{OH}	Output Voltage Level • Logic Low I_{LOAD} = 500µA • Logic High I_{LOAD} = - 500µA	V_{CC} = 4.75V	 2.7		0.4	V
	Clock Input Voltage Level • Logic Low • Logic High	V_{CC} = 5 ± 0.25V	 2.5		0.6	V
	Input Capacitance	V_{offset} = 2.5V, F = 1MHz			10	pF

3220-07.TBL

AC ELECTRICAL CHARACTERISTICS

Operating conditions : V_{SS} = 0V, T_A = 0 to 70°C, V_{CC} = 5V ± 5%, unless otherwise specified
Output Loads : Capacitance = 50pF, Current Logic Low = 500µA

TEST LOAD ON OUTPUTS



3220-25.EPS

7. TIMING DIAGRAMS

TIMING PARAMETERS

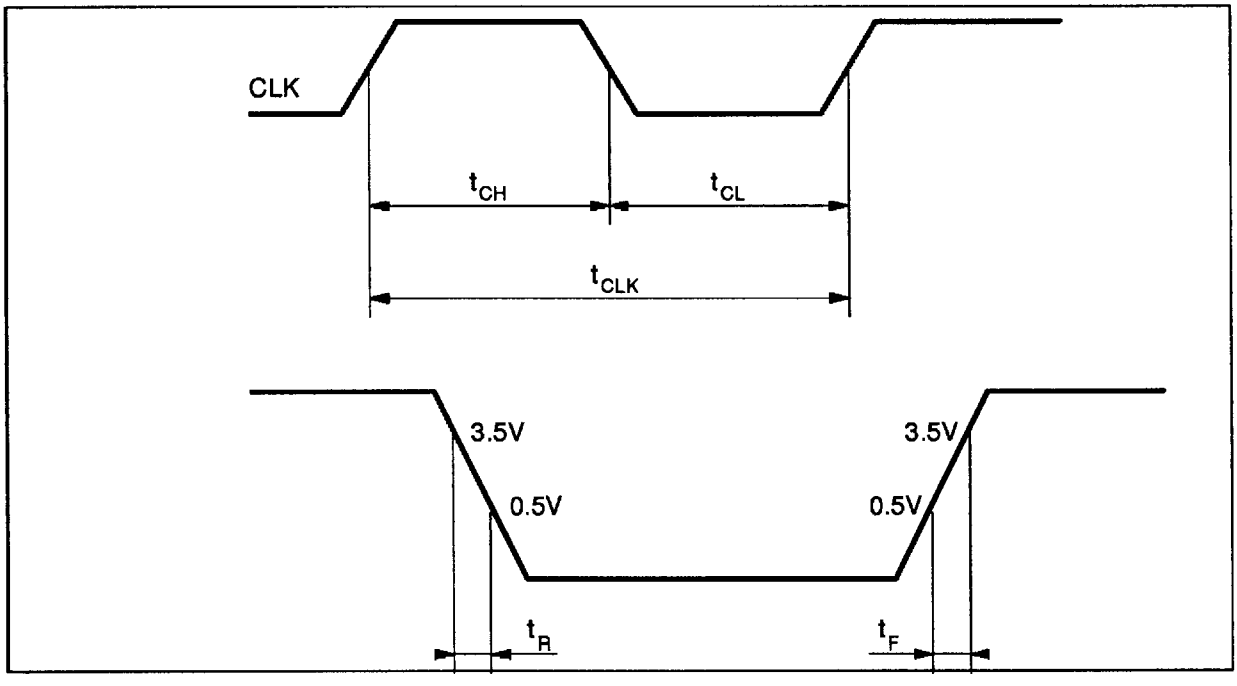
Symbol	Parameter	Min.	Typ.	Max.	Unit
t_{CH}	Clock Pulse High Width	20			ns
t_{CL}	Clock Pulse Low Width	20			ns
t_{CLK}	Clock Period	50			ns
t_R	Clock Rise Time (see note)	0		10	ns
t_F	Clock Fall Time (see note)	0		10	ns
t_{SDCL}	Data Setup Time from CLK ↑	8			ns
t_{HDCL}	Data Hold Time from CLK ↑	0			ns
t_{DO}	Output Data Delay from CLK ↑			20	ns
t_{EN1}	Enable Hold Time from CLK ↑	0			ns
t_{EN2}	Enable Rising Edge Setup Time from CLK ↓	5			ns
t_{EN3}	Enable Falling Edge Setup Time from CLK ↑	30			ns
t_{OFF}	Delay from OE ↓ to Output going to High Impedance			20	ns
t_{ON}	Delay from OE ↑ to Output going to High Impedance			20	ns
t_{OE1}	CLK Rising Edge to OE going Low or High	0			ns
t_{OE2}	OE Setup Time	8			ns

3220-06.TBL

Note : The clock edges should be monotonic between V_{IL} and V_{IH}.

TIMING WAVEFORMS

Figure 24 : Clock Timing Waveform



3220-06.EPS

Figure 25 : Data Timing Waveforms

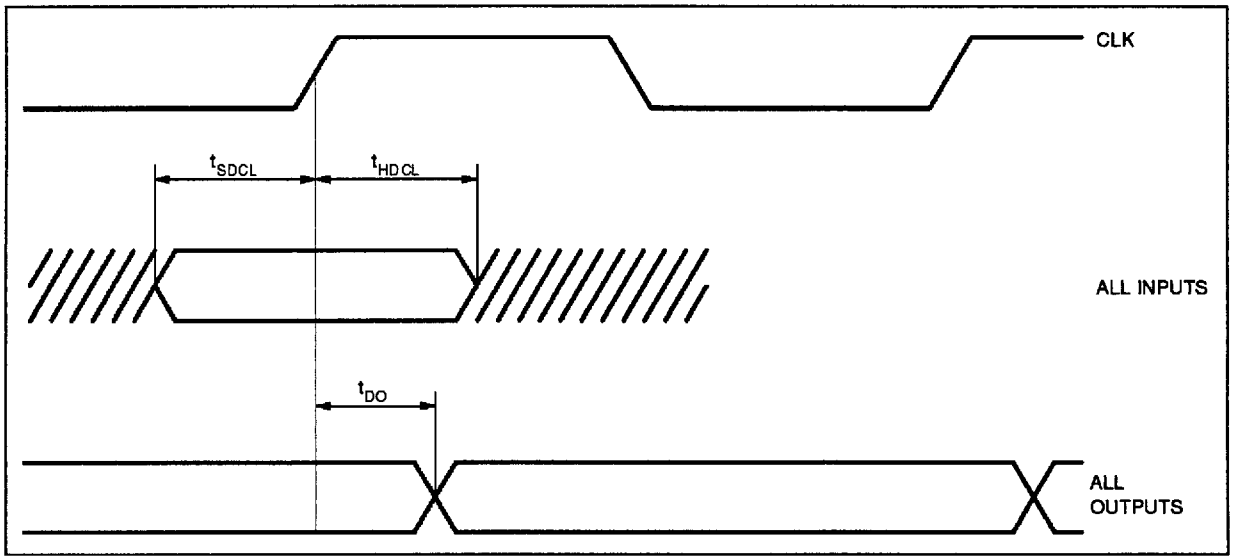
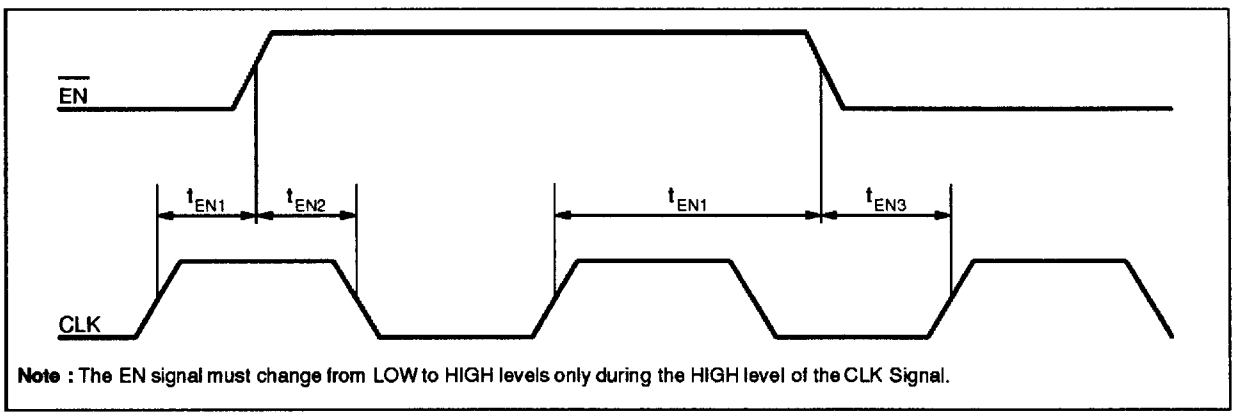
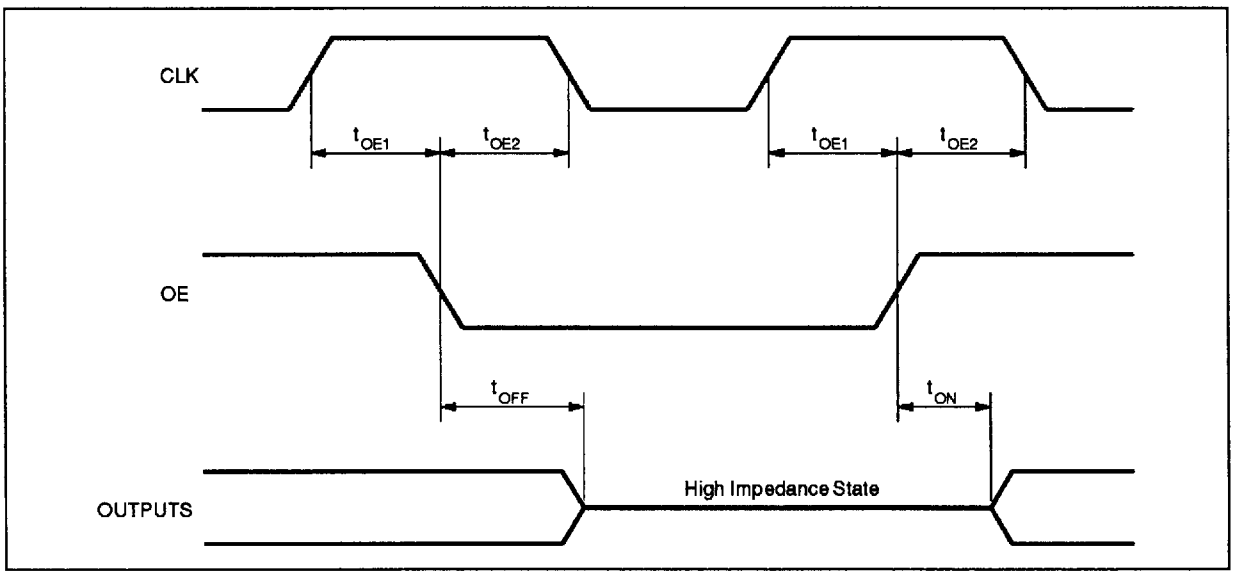


Figure 26 : Clock Enable Timing Waveform

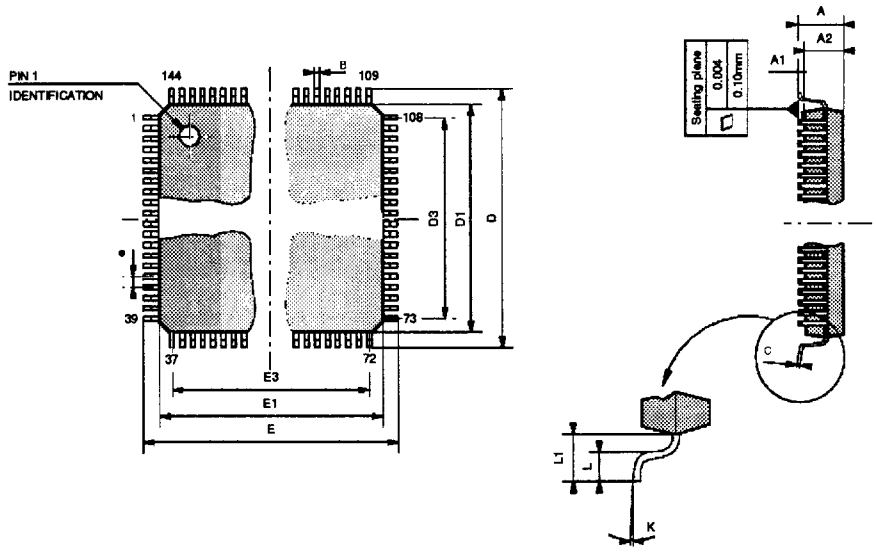


Note : The EN signal must change from LOW to HIGH levels only during the HIGH level of the CLK Signal.

Figure 27 : Output Enable Timing Waveform



PACKAGE MECHANICAL DATA
144 PINS - PLASTIC QUAD FLAT PACK



PMPQF144.EPS

Dim.	mm			inches		
	Min	Typ	Max	Min	Typ	Max
A			4.07			0.160
A1	0.25			0.010		
A2	3.17	3.42	3.67	0.125	0.133	0.144
B	0.22		0.38	0.009		0.015
C	0.13		0.23	0.005		0.009
D	30.95	31.20	31.45	1.219	1.228	1.238
D1	27.90	28.00	28.10	1.098	1.102	1.106
D3		22.75			0.896	
e		0.65			0.026	
E	30.95	31.20	31.45	1.219	1.228	1.238
E1	27.90	28.00	28.10	1.098	1.102	1.106
E3		22.75			0.896	
L	0.65	0.80	0.95	0.026	0.031	0.037
L1		1.60			0.063	
K	0^o (min.), 7^o (max.)					

PQFP144.TBL

SGS-THOMSON MICROELECTRONICS