SENSORAY CO., INC.

PC/104 MPEG-1/2 VIDEO CODEC BOARD

Model 512

October 19, 2001

7313 SW Tech Center Dr.
Tigard, OR 97223
Phone 503.684.8005 • Fax 503.684.8164
sales@sensoray.com
www.sensoray.com

TABLE OF CONTENTS

1. Limited Warranty

Sensoray Company, Incorporated (Sensoray) warrants the model 512 hardware to be free from defects in material and workmanship and perform to applicable published Sensoray specifications for two years from the date of shipment to purchaser. Sensoray will, at its option, repair or replace equipment that proves to be defective during the warranty period. This warranty includes parts and labor.

The warranty provided herein does not cover equipment subjected to abuse, misuse, accident, alteration, neglect, or unauthorized repair or installation. Sensoray shall have the right of final determination as to the existence and cause of defect.

As for items repaired or replaced under warranty, the warranty shall continue in effect for the remainder of the original warranty period, or for ninety days following date of shipment by Sensoray of the repaired or replaced part, whichever period is longer.

A Return Material Authorization (RMA) number must be obtained from the factory and clearly marked on the outside of the package before any equipment will be accepted for warranty work. Sensoray will pay the shipping costs of returning to the owner parts which are covered by warranty.

Sensoray believes that the information in this manual is accurate. The document has been carefully reviewed for technical accuracy. In the event that technical or typographical errors exist, Sensoray reserves the right to make changes to subsequent editions of this document without prior notice to holders of this edition. The reader should consult Sensoray if errors are suspected. In no event shall Sensoray be liable for any damages arising out of or related to this document or the information contained in it.

EXCEPT AS SPECIFIED HEREIN, SENSORAY MAKES NO WARRANTIES, EXPRESS OR IMPLIED, AND SPECIFICALLY DISCLAIMS ANY WARRANTY OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. CUSTOMER'S RIGHT TO RECOVER DAMAGES CAUSED BY FAULT OR NEGLIGENCE ON THE PART OF SENSORAY SHALL BE LIMITED TO THE AMOUNT THERETOFORE PAID BY THE CUSTOMER. SENSORAY WILL NOT BE LIABLE FOR DAMAGES RESULTING FROM LOSS OF DATA, PROFITS, USE OF PRODUCTS, OR INCIDENTAL OR CONSEQUENTIAL DAMAGES, EVEN IF ADVISED OF THE POSSIBILITY THEREOF.

2. Special Handling Instructions

The Model 512 board contains CMOS circuitry that is sensitive to Electrostatic Discharge (ESD). Special care should be taken in handling, transporting, and installing the 512 to prevent ESD damage to the board. In particular:

- Do not remove the 512 from its protective antistatic bag until you are ready to install it in your computer.
- Handle the 512 only at grounded, ESD protected stations.
- Always turn off the computer before installing or removing the 512 board

3. Introduction

The Sensoray Model 512 is an MPEG video encoder decoder board. Some of the feature include:

General

- Real time MPEG-2 and MPEG-1 video encoder and decoder
- Support for variable bit rate and constant bit rate
- IPB pictures to 15Mbps for constant bit rate and 10Mbps for variable bit rate
- Supports multiple resolutions (704x480, 640x480, 352x240, etc.)
- Support for NTSC, PAL
- Composite and Svideo inputs and outputs
- During encoding and standby, video input is fed to output for easy adjustment
- DOS, Linux, and Windows driver available
- PC/104 form factor
- Either BNC connectors or header for video input and output
- Either mini phone jacks or header for audio input and output
- 3 digital TTL level digital inputs and/or outputs
- Low power

Video encoder

- Generates 13818 (MPEG-2) and 11172 (MPEG-1) compliant elementary streams (ES)
- Operates up to 30 frames per second
- Selectable bit rate

Video decoder

- Decodes both MPEG-1 and MPEG-2 streams
- Horizontal and vertical scaling

DOS Driver features

- Low memory usage
- All functions called through the i86 software interrupt mechanism for universal language support
- Easy calls to functions that return or send blocks of compressed video to and from the 512 hardware
- Easy calls to functions that return or send blocks of uncompressed audio to and from the 512 hardware
- Status functions
- Digital I/O functions
- Command line utility to set up video parameters such as bit rate, resolution, etc.

4. Hardware Installation

The 512 is installed on a PC/104 stack. Four mounting holes are provided, one in each corner, for mechanical stability. The bus connector are the large ones as seen on the bottom of figure 1.

This base address of the 512 must be selected using the base address selection jumper, as seen in the lower left hand corner of figure 1. Address selection will be discussed in a following section.

Video and digital I/O are present on five connectors. The header, J2, has all I/O signals, including Svideo, composite video, and digital. J1, J3 are for composite video only.

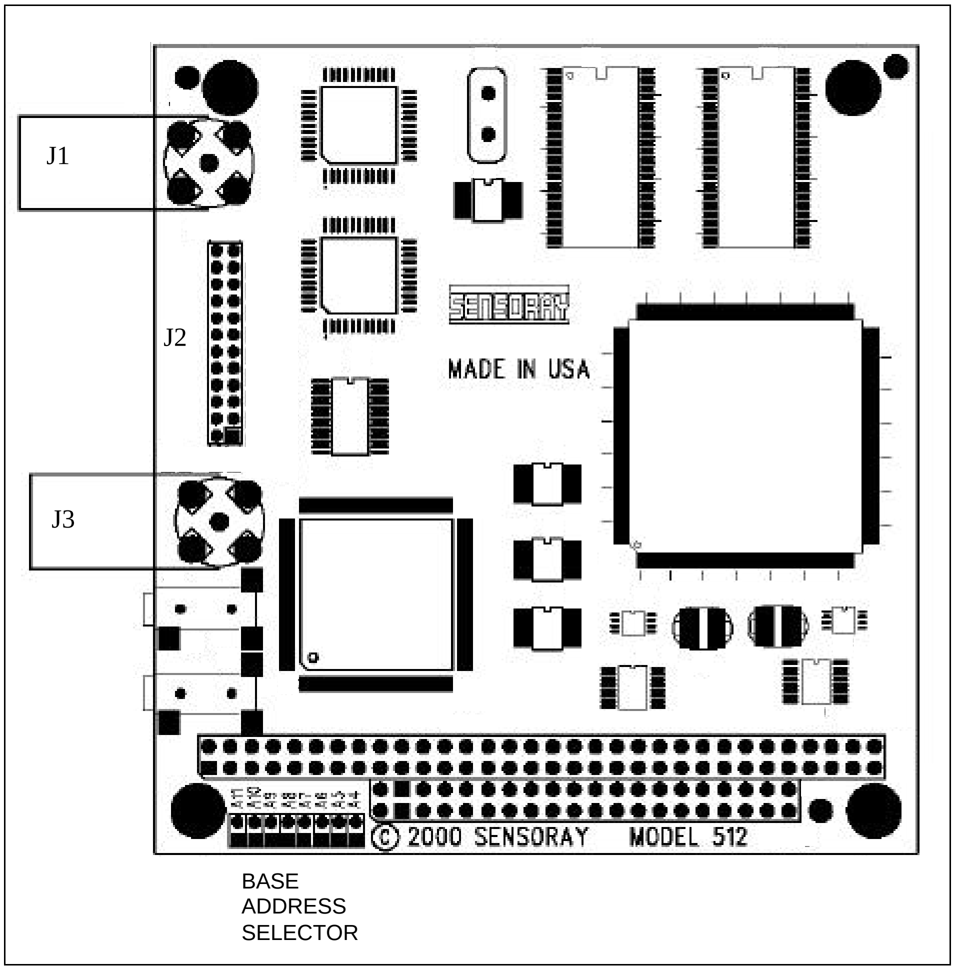

Figure 1 Board Layout

4.1 Base address selection

The jumpers marked A6 through A11 are for address selection. A “1” in the table below indicates an installed jumper. A ‘0’ is no jumper. Using the table the user can determine the base address of the 512 in hexadecimal.

A11	A10	A9	A8	First digit of address		A7	A6	Second digit of address		Third digit of address
0	0	0	0	F		0	0	C		0
0	0	0	1	E		0	1	8		
0	0	1	0	D		1	0	4		
0	0	1	1	C		1	1	0		
0	1	0	0	B						
0	1	0	1	A						
0	1	1	0	9						
0	1	1	1	8						
1	0	0	0	7						
1	0	0	1	6						
1	0	1	0	5						
1	0	1	1	4						
1	1	0	0	3						
1	1	0	1	2						
1	1	1	0	1						
1	1	1	1	0						

Table 1 Base Address Selection

Example 1:

A11, A10 not installed, A9, A8 installed (First digit is C)
A7,A6 not installed (Second digit is C)
The third digit is always zero, hence we have a base address of CC0 hex

Example 2:

We need a base address of F80 hex.
The first digit is F, therefore jumpers A11,A10,A9,A8 must be removed.
The second digit is 8, therefore jumper A7 must be removed and jumper A6 installed.

If you have two 512’s in your system they must have different base addresses. Be sure that you are not using an address used by another device in your system.

VERY IMPORTANT: For normal operation a jumper must be installed at position A5 and no jumper installed at position A4.

4.2 Connectors

The header J2 carries all the I/O signals. The pinout is given in Table 2.

Pin	Function	Pin	Function
1	Digital I/O 2	2	+5V
3	Digital I/O 1	4	Digital ground
5	Digital I/O 0	6	Digital ground
7	N/A	8	N/A
9	N/A	10	N/A
11	SVideo out – Y	12	Video ground
13	SVideo out – C	14	Video ground
15	Composite video out	16	Video ground
17	Composite video in 4 / SVideo 1 – C	18	Video ground
19	Composite video in 3 / SVideo 2 – C	20	Video ground
21	Composite video in 2 / SVideo 1 – Y	22	Video ground
23	Composite video in 1 / SVideo 2 – Y	24	Video ground

Table 2 Header J1 Pinout

The BNC connector (if installed) are for composite video. J1 is composite video out and J3 is composite video in 1. The center pin is the signal and the out shell is video ground.

5. DOS Driver

Using the DOS COPY command, copy the files from the distribution diskette to your system. It is a good idea to create a subdirectory for the files. For example,

```
MD C:\X12DOS
COPY A:\DOS\*.* C:\X12DOS
```

Before an application can use the driver the TSR (terminate and stay resident) executable X12TSR.EXE

This is used to setup bit rate, resolution, etc.

A

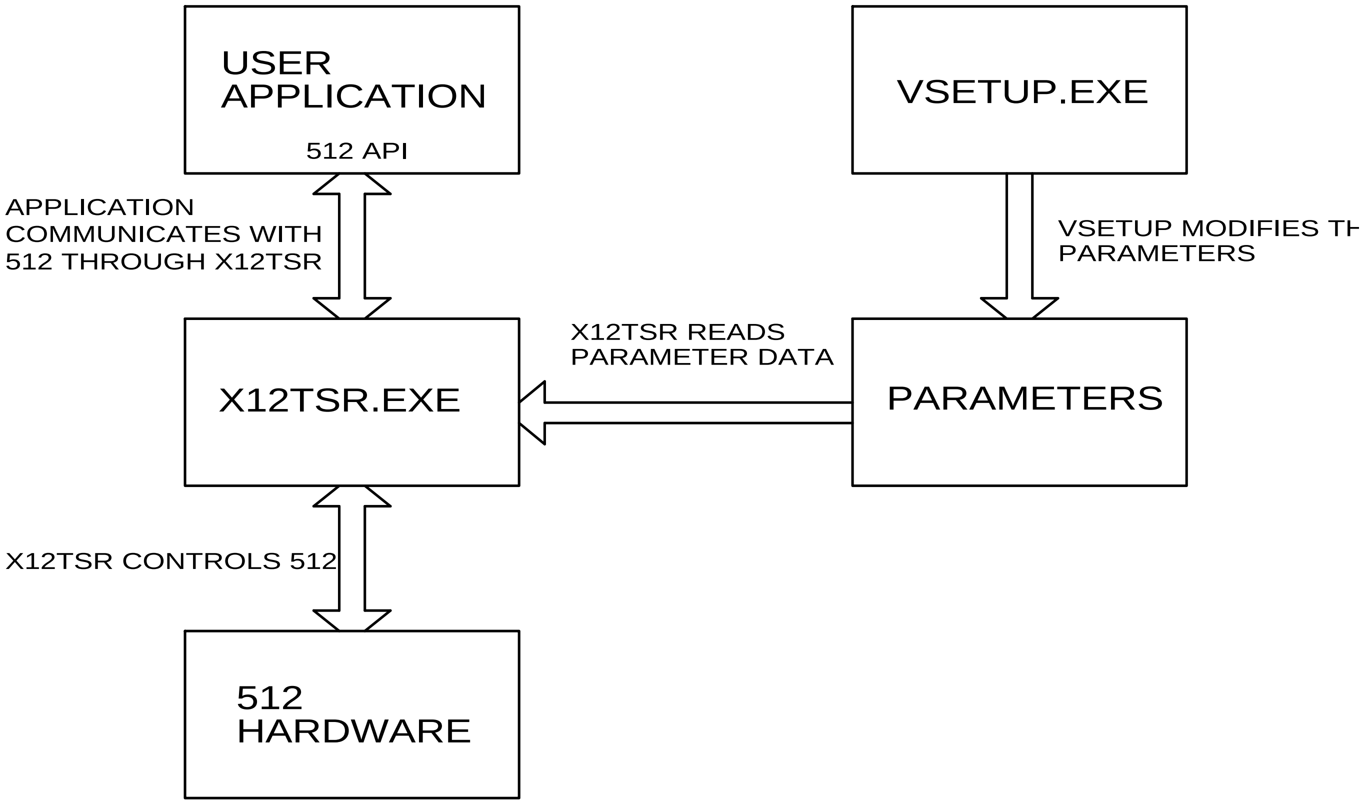

Figure 2

X12TSR.EXE. An API (application program interface) is provided to facilitate 512 application development. The API provides commands to start/stop the MPEG encoder and decoder, commands to retrieve or send MPEG data to and from the 512, commands to obtain 512 encoder/decoder status, and commands to control the digital I/O signals. The parameter data which is maintained by VSETUP.EXE is read by X12TSR.EXE when acquisition and compression (i.e. encoding) are initiated by the application. The video system parameter is also read by X12TSR.EXE when X12TSR.exe is initially loaded. A typical system might first set the path to the system files (see section 5.1), then run VSETUP.EXE to set the required parameters, load X12TSR.EXE, run the application, etc.

5.1 X12TSR.EXE

X12TSR provides the interface to the 512 hardware and must be loaded before an application can use the 512. The x86 software interrupt mechanism is used by an application to communicate with the driver and allows the application to be language independent. In addition to providing the driver functions, it also initializes the video encoder and decoder and loops the incoming video to the video output.

Before loading X12TSR.EXE the driver must know the location of all the parameter files (*.hex, *.bin, etc.) If the files are in the current directory nothing needs to be done. If the files are not in the current directory an environment variable X12PATH must be set to inform the driver about the location of the various parameter files.

If the environment variable is to be used, the statement below could be added to your AUTOEXEC.BAT file.

```
SET X12PATH=<driver files directory>.
```

For example,

```
SET X12PATH=C:\X12DOS
```

X12TSR.EXE is loaded by typing X12TSR at the DOS prompt or by adding it to the AUTOEXEC.BAT file for automatic loading at boot time. A number of command line switches are provided to ensure proper initialization of the driver. The command line switches are described below.

```
C:\path>X12TSR [HELP][-2][-Axxx][-Sx][-Cx][-RM][-NI] [[-Axxx][-Sx][-Cx]]
```

Where,

HELP	Displays a list of the switches. Does not load TSR.
-2	Indicates that there are two 512s in your system. Do not use if there is only one 512.
-Axxx	Sets 512 base address (hex, default CC0 for first 512, DC0 for second).
-Sx	Sets video source input to SVideo input x (1-2).
-Cx	Sets video source input to composite input x (1-4, default input 1).
-RM	Removes 512 driver from memory, if possible.
-NI	'No-install' initialized 512 video without installing driver.

NOTE: All parameters between [ ] are optional. Default values will be provided if not used.

NOTE: If there are two 512s in system, the -A, -S, and -C switches can be repeated on the command line for the second 512.

Example(1),

```
SET X12PATH =C:\X12DOS
C:\X12DOS\X12TSR
```

Loads X12TSR.EXE for one 512 and uses the defaults (i.e. the video input is composite video 1, base address of 512 is CC0 (hex),

Example(2),

```
SET X12PATH =C:\X12DOS
C:\X12DOS\X12TSR –S1 –AF00
```

Loads X12TSR.EXE for one 512 and sets the video input to SVideo input 1 and the base address of 512 to F00 (hex).

Example(3),

```
SET X12PATH =C:\X12DOS
```

```
C:\X12DOS\X12TSR  -2 –S1 –C1 –AF00 –AC00
```

Loads X12TSR.EXE for two 512s and sets the video input to SVideo input 1 and the base address to F00 (hex) for the first 512, and composite video input 1 and the base address to C00 (hex) for the second.

NOTE: When shipped, the parameters files are setup for NTSC and video will not pass through correctly if a PAL source is connected. To fix the problem run VSETUP.EXE with the –PAL switch and reload the TSR. The 512 will now remain in PAL mode until VSETUP.EXE is run again with the –NTSC switch. Before you can reload the TSR you must either reboot your machine or type X12TSR –RM to unload the TSR from system memory.

5.2 VSETUP.EXE

VSETUP.EXE is run from the command line and is used to modify the video and compression parameters of the 512. Once modified, the parameters will remain in effect, even through power cycling, until they are changed again by VSETUP.EXE. Parameters can be modified at any time and take effect when X12TSR.EXE is loaded or when an application starts the encode or decoder. VSETUP.EXE has the following command line syntax.

```
C:\path>VSETUP [-1/-2][-B][-NTSC][-PAL][-Rxx][-Wxxx][-Hxxx][-M1][-M2][-VON][-VOFF][-Z][-Q]
```

Command line switches:

-1/-2	selects 512 board for which the parameters are to be modified. I.e. 1 (default) or 2.
-NTSC/-PAL	selects the video system used. I.e. NTSC or PAL. Default: NTSC
-Gxx	sets number of video frames in a MPEG GOP. Valid numbers are 3, 6, 9, 12, or 15. Default: 15
-Rxx	sets the desired compression bit rate in millions of bits per second (Mbps). i.e. -R12 for 12Mbps, -R0.5 for 0.5Mbps, etc. Default: 4 Mbps
-VON/-VOFF	-VON enable variable bit rate compression, -VOFF enables constant bit rate compression. Default: VOFF

The following switches set the compression system as well as the capture resolution. (Default: P0)

-P0	MPEG-2 compression,	Size D1:	NTSC - 704x480, PAL - 704x576 pixels
-P1	MPEG-2 compression,	Size 1/2 D1:	NTSC - 352x480, PAL - 352x576 pixels
-P2	MPEG-2 compression,	Size 2/3 D1:	NTSC - 480x480, PAL - 480x576 pixels
-P3	MPEG-1 compression,	Size CIF:	NTSC - 352x240, PAL - 352x288 pixels
-M1	same as -P3		
-M2	same as -P0		

-Z	loads default values for all parameters.
-Q	quiet mode.

NOTE: Using VSETUP with no switches displays the current settings as well as a list of available switches.

NOTE: VSETUP.EXE only changes the parameter(s) selected to be modified. All other parameters will remain unchanged.

Example,

```
C:\X12DOS\VSETUP -Z
C:\X12DOS\VSETUP -PAL  -M1
```

The first line loads the default parameters, the second line changes the video system to PAL and the compression system to MPEG1. The 512 is setup as follows:

Bit rate is 4Mbps
Constant bit rate
PAL video
MPEG1 compression
GOP of 15
352x288 capture resolution

If we now execute the following:

```
C:\X12DOS\VSETUP –R6
```

We have:

Bit rate is 6Mbps
Constant bit rate
PAL video
MPEG1 compression
GOP of 15
352x288 capture resolution

6. The 512 Driver Interface

All the functions of the 512 DOS device driver are called via the x86 software interrupt mechanism. The interrupt number used for all the SDK functions is **X12_INT**. To use the functions, the file SX12.H must be included in the application source files.

Example:

```
#include “sx12.h”                                          //Included for 512 dos driver support
#include “dos.h”                                           //Included for software interrupt support
.
.
.
X12PARAM _far *Param;                                      //Declare a pointer to an X12PARAM object
union REGS regs;                                           //
.
.
.
regs.x.si =  X12_GETCMDPARAPTR + CARD0;                   //Select the driver function
int86(X12INT,&regs,&regs);                                 //Call the driver
Param = (X12PARAM _far *) MK_FP(regs.x.dx, regs.x.ax);   //Make a pointer from the returned register values
```

The 512 driver has two function calls. The first, **X12_GETCMDPARAPTR**, gets a pointer to a parameter structure and the second, **X12_SENDCOMMAND**, sends a command to the 512 driver and board.

6.1 X12PARAM structure

```
typedef struct
{
    unsigned short Mode;
    short Reserved1;
    unsigned long VQSize;
    short Input;
    short InputType;
    short Reserved4;
    unsigned short DI0;
    unsigned short DI1;
    unsigned short DI2;
    unsigned short DO0;
    unsigned short DO1;
    unsigned short DO2;
    unsigned short DDIR0;
    unsigned short DDIR1;
    unsigned short DDIR2;
    short Reserved5;
    short Reserved6;
    unsigned short DroppedFrames;
    long Length;
    unsigned long FrameCount;
    short Status;
    void _far *Buffer;
} X12PARAM;
```

Member	Description
Mode	This sets the 512 mode and must be set prior to issuing a **X12_VI_START** command.
VQSize	Current video queue size and is updated after any command including **X12_VI_GETSTATUS.**
Input	This is set to the desired video input prior to issuing a **X12_VI_SETINPUT** command.
InputType	This is set to the desired video input type prior to issuing a **X12_VI_SETINPUTTYPE** command.
DI0, DI1, DI2	Returns the state of digital I/O pin 0, 1, or 2**.** Updated after any command including **X12_VI_GETSTATUS.**
DO0, DO1, DO2	Sets the state of digital I/O pin 0, 1, or 2**.** For the setting to take effect the pin must also be set to output. The outputs are updated after any command including **X12_VI_GETSTATUS.**
DDIR0, DDIR1, DDIR2	Sets the direction of digital I/O pin 0, 1, or 2. The directions are updated after any command including **X12_VI_GETSTATUS.**

Mode:

Value	Description
ENCODE	Enables encoding mode (compression)
DECODE	Enables decoding mode (decompression)

Input:

Value	Description
VIN1	Video input 1
VIN2	Video input 2
VIN3	Video input 3 (composite video only)
VIN4	Video input 4 (composite video only)

InputType:

Value	Description
CVIDEO	Input is composite video
SVIDEO	Input is svideo

DI0, DI1, DI2:

Value	Description
HIGH	Pin input is at 5V
LOW	Pin input is at 0V

DO0, DO1, DO2:

Value	Description
HIGH	Pin output is at 5V
LOW	Pin output is at 0V

DDIR0, DDIR1, DDIR2:

Value	Description
INPUT	Pin is input
OUTPUT	Pin is output

HINT: If an open collector style output is desired, set the desired output to **LOW** and use the direction control to set the output high or low. E.g. if **DO0** is set to **LOW** and **DDIR0** is set to **OUTPUT** digital I/O pin 0 will go to 0V and if **DDIR0** is then set to **INPUT** digital I/O pin 0 will go to 5V.

DroppedFrames	Contains the number of frames that could not be compressed. When the video queue is not emptied at a sufficiently high rate, the compression engine stalls and begins dropping frames. This value is update after any command including **X12_VI_GETSTATUS**.
Length	Length of MPEG buffer data to be transferred to/from the video queue. **Length** must be set before executing either **X12_VI_GETDMADATA** or **X12_VI_PUTDMADATA**. After the execution of these commands **Length** contains the actual number of bytes transferred.
FrameCount	Contains the number of frames compressed during encoding. This value is updated after any command including **X12_VI_GETSTATUS**.
Status	After the execution of a command **Status** should be checked to verify that the command was executed successfully.

Value	Description
SUCCESS	Command completed successfully
FAIL	Command failed

Buffer	This is a far pointer (32-bit real mode ‘segment:offset’ pair) to the local MPEG buffer. The buffer is used to transfer MPEG to/from the 512 video queue.

6.2 X12_GETCMDPARAPTR function call

The **X12_GETCMDPARAPTR** function obtains a real mode far pointer to the parameter data structure. A far pointer to the data permits the sending and returning of values to and from the driver pertinent to a particular command.

Entry:

Register	Description	Value
SI	Function number	**X11_ GETCMDPARAPTR** + either **CARD0** for the first card or **CARD1** for the second card.

Exit:

Register	Description	Value
DX:AX	Segment-offset pair pointer to parameter data.	

Example:

```
#include “x12.h”                                    //Included for 512 dos driver support
#include “dos.h”                                    //Included for software interrupt support
.
.
.
X12PARAM _far *Param;                               //Declare a pointer to an X12PARAM object
union REGS regs;
.
.
.
regs.x.si =  X12_GETCMDPARAPTR + CARD0;             //Select the driver function
int86(X12INT,&regs,&regs);                          //Call the driver
Param = (X12PARAM _far *) MK_FP(regs.x.dx, regs.x.ax);  //Make a pointer from the returned register values
```

6.3 X12_SENDCOMMAND function call

The **X12_SENDCOMMAND** function is used to send a command to the 512 driver. See section 6.3 for command details.

Entry:

Register	Description	Value
SI	Function number	**X12_SENDCOMMAND** + either **CARD0** for the first card or **CARD1** for the second card.
AX	Command	**X12_VI_START** **X12_VI_STOP** **X12_VI_GETDMADATA** **X12_VI_PUTDMADATA** **X12_VI_SETINPUT** **X12_VI_SETINPUTTYPE** **X12_VI_GETSTATUS** **X12_RESETDEVICE**

Exit:

Register	Description	Value
AX	Timeout flag	Non-zero is timeout Zero is success

Example:

```
#include “x12.h”                                          //Included for 512 dos driver support
#include “dos.h”                                          //Included for software interrupt support
.
.
X12PARAM _far *Param;                                     //Declare a pointer to an X12PARAM object
union REGS regs;                                          //
.
.
regs.x.si =  X12_GETCMDPARAPTR + CARD0;                   //Select the driver function
int86(X12INT,&regs,&regs);                                //Call the driver
Param = (X12PARAM _far *) MK_FP(regs.x.dx, regs.x.ax);    //Make a pointer from the returned register values

Param->Mode = ENCODE;
regs.x.ax =  X12_VI_START;                                //The video start command
regs.x.si =  X12_SENDCOMMAND + CARD0;                     //Select the driver function
int86(X12INT,&regs,&regs);                                //Call the driver

if(regs.x.ax)
{
      printf("Error Timeout\n");
      return 0;
}
```

IMPORTANT: Every time the **X12_SENDCOMMAND** is called any changes to the **X12PARAM** members **DO0, DO1, DO2** or **DDIR0, DDIR1, DDIR2** take effect. Upon return **VQSize**, **DI0**, **DI1**, **DI2**, **DroppedFrames**, **FrameCount**, and **Status** are updated.

6.4 The 512 commands

The command described in this section are used with the **X12_SENDCOMMAND** function call (see section 6.2). These commands use the **X12PARAM** structure to pass parameters and return results.

6.4.1 X12_VI_START

This command starts video only compression or decompression.

Entry:

Parameter	Type	Value
Mode	**unsigned short**	**ENCODE** or **DECODE**

Exit:

Parameter	Type	Value
Status	**short**	**SUCCESS** or **FAIL**

Example:

```
Param->Mode = ENCODE;                                    //Compression
regs.x.ax =  X12_VI_START;
regs.x.si =  X12_SENDCOMMAND + CARD0;                    //Select the driver function
int86(X12INT,&regs,&regs);                               //Call the driver

Param->Mode = DECODE;                                    //Decompression
regs.x.ax =  X12_VI_START;
regs.x.si =  X12_SENDCOMMAND + CARD0;                    //Select the driver function
int86(X12INT,&regs,&regs);                               //Call the driver
```

Note: In addition to the above mentioned parameters, any changes to the **X12PARAM** members **DO0, DO1, DO2** or **DDIR0, DDIR1, DDIR2** take effect when this command is called. Upon return **VQSize**, **DI0**, **DI1**, **DI2**, **DroppedFrames**, and **FrameCount** are updated.

6.4.2 X12_VI_STOP

This command stops video compression or decompression only.

Entry:

Parameter	Type	Value
None		

Exit:

Parameter	Type	Value
Status	**short**	**SUCCESS** or **FAIL**

Example:

```
regs.x.ax =  X12_VI_STOP;
regs.x.si =  X12_SENDCOMMAND + CARD0;           //Select the driver function
int86(X12INT,&regs,&regs);                      //Call the driver
```

Note: In addition to the above mentioned parameters, any changes to the **X12PARAM** members **DO0, DO1, DO2** or **DDIR0, DDIR1, DDIR2** take effect when this command is called. Upon return **VQSize**, **DI0**, **DI1**, **DI2**, **DroppedFrames**, and **FrameCount** are updated.

6.4.3 X12_VI_GETDMADATA

This command gets raw compressed video data from the video DMA buffer (video queue). The data is ES (elementary stream) MPEG encoded.

Entry:

Parameter	Type	Value
Buffer	**far** pointer	Pointer to a buffer allocated by the user application for video data.
Length	**long**	Number of bytes of MPEG data desired

Exit:

Parameter	Type	Value
Status	**short**	**SUCCESS** or **FAIL. SUCCESS** is returned when there are more than zero bytes of MPEG data available.
Length	**long**	Number of bytes of data actually read from the queue
FrameCount	**unsigned long**	Number of video frames compressed since the start of compression.

Example:

```
BYTE _far Buffer[VIDEO_BUFFER_SIZE];

Param->Buffer = Buffer;
Param->Length = VIDEO_BUFFER_SIZE;
regs.x.ax =  X12_VI_GETDMADATA;
regs.x.si =  X12_SENDCOMMAND + CARD0;           //Select the driver function
int86(X12INT,&regs,&regs);                      //Call the driver

if(Param->Status == SUCCESS)                    //if successful, write the data of length Param->Length
{                                               //to a file
    write(fhandle, Buffer, Param->Length);
}
```

Note: In addition to the above mentioned parameters, any changes to the **X12PARAM** members **DO0, DO1, DO2** or **DDIR0, DDIR1, DDIR2** take effect when this command is called. Upon return **VQSize**, **DI0**, **DI1**, **DI2**, **DroppedFrames**, and **FrameCount** are updated.

6.4.4 X12_VI_PUTDMADATA

This command puts ES (elementary stream) MPEG encoded video data into the video DMA buffer (video queue).

Entry:

Parameter	Type	Value
Buffer	**far** pointer	Pointer to a buffer allocated by the user application holding video data.
Length	**long**	Number of bytes of MPEG data to be decompressed

Exit:

Parameter	Type	Value
Status	**short**	**SUCCESS** or **FAIL. FAIL** is returned if data was not Written.
Length	**long**	Number of bytes of MPEG data actually written.

Example:

```
BYTE _far Buffer[VIDEO_BUFFER_SIZE];

Param->Mode = DECODE;
regs.x.ax  =  X12_VI_START;
regs.x.si  =  X12_SENDCOMMAND + CARD0;              //Select the driver function
int86(X12INT,&regs,&regs);                          //Call the driver

Param->Length = read(fhan,Buffer, VIDEO_BUFFER_SIZE);

while(!kbhit())
{
    Param->Buffer = Buffer;
    regs.x.ax = X12_VI_PUTDMADATA;
    regs.x.si =  X12_SENDCOMMAND + CARD0;           //Select the "Send command' driver function
    int86(X12INT,&regs,&regs);                      //Call the driver

    if(Param->Status == SUCCESS)
      if((Param->Length = read(fhan,Buffer, Param->Length)) == EOF)
        break;
}
```

Note: In addition to the above mentioned parameters, any changes to the **X12PARAM** members **DO0, DO1, DO2** or **DDIR0, DDIR1, DDIR2** take effect when this command is called. Upon return **VQSize**, **DI0**, **DI1**, **DI2**, **DroppedFrames**, and **FrameCount** are updated.

6.4.5 X12_VI_SETINPUT

This command selects the video input.

Entry:

Parameter	Type	Value
Input	**short**	**VIN1** for Video input 1 **VIN2** for Video input 2 **VIN3** for Video input 3 (composite video only) **VIN4** for Video input 4 (composite video only)

Exit:

Parameter	Type	Value
Status	**short**	**SUCCESS** or **FAIL**

Example:

```
X12PARAM _far *Param;                          //Declare a pointer to an X12PARAM object
union REGS regs;
.
.
Param->Input = VIN2;
regs.x.ax  =  X12_VI_SETINPUT;
regs.x.si  =  X12_SENDCOMMAND + CARD0;         //Select the driver function
int86(X12INT,&regs,&regs);                     //Call the driver
```

Note: In addition to the above mentioned parameters, any changes to the **X12PARAM** members **DO0, DO1, DO2** or **DDIR0, DDIR1, DDIR2** take effect when this command is called. Upon return **VQSize**, **DI0**, **DI1**, **DI2**, **DroppedFrames**, and **FrameCount** are updated.

6.4.6 X12_VI_SETINPUTTYPE

This command selects the video input type.

Entry:

Parameter	Type	Value
InputType	**short**	**CVIDEO** for composite video **SVIDEO** for svideo

Exit:

Parameter	Type	Value
Status	**short**	**SUCCESS** or **FAIL**

Note: This command should be executed before the input number is selected, if an input other that input 1 is to be used.

Example:

```
X12PARAM _far *Param;                        //Declare a pointer to an X12PARAM object
union REGS regs;
.
.
Param->InputType = SVIDEO;
regs.x.ax  =  X12_VI_SETINPUTTYPE;
regs.x.si  =  X12_SENDCOMMAND + CARD0;       //Select the driver function
int86(X12INT,&regs,&regs);                   //Call the driver

Param->Input = VIN1;
regs.x.ax  =  X12_VI_SETINPUT;
regs.x.si  =  X12_SENDCOMMAND + CARD0;       //Select the driver function
int86(X12INT,&regs,&regs);                   //Call the driver
```

Note: In addition to the above mentioned parameters, any changes to the **X12PARAM** members **DO0, DO1, DO2** or **DDIR0, DDIR1, DDIR2** take effect when this command is called. Upon return **VQSize**, **DI0**, **DI1**, **DI2**, **DroppedFrames**, and **FrameCount** are updated.

6.4.7 X12_VI_GETSTATUS

This command updates the **X12PARAM** members. Changes to the **X12PARAM** members **DO0, DO1, DO2** or **DDIR0, DDIR1, DDIR2** take effect when this command is called. Upon return **VQSize**, **DI0**, **DI1**, **DI2**, **DroppedFrames**, and **FrameCount** are updated.

Exit:

Parameter	Type	Value
Status	**short**	**SUCCESS** or **FAIL**

6.4.8 X12_RESETDEVICE

This command resets the 512 MPEG engine.

Entry:

Parameter	Type	Value

Exit:

Parameter	Type	Value
Status	**short**	**SUCCESS** or **FAIL**

Note: In addition to the above mentioned parameters, any changes to the **X12PARAM** members **DO0, DO1, DO2** or **DDIR0, DDIR1, DDIR2** take effect when this command is called. Upon return **VQSize**, **DI0**, **DI1**, **DI2**, **DroppedFrames**, and **FrameCount** are updated.

Appendix A: Quick setup and demo

1) Turn all power off.
2) Place 512 address selection jumpers on positions A8, A9 and A5. Base address CC0.
3) Place 512 on PC/104 stack.
4) Connect video camera to J3.
5) Connect video monitor to J1.
6) Turn power on.
7) At command prompt type: **C:\> MD \X12DOS** and press return. This creates a directory for the 512 DOS driver files.
8) Insert distribution diskette in floppy drive.
9) At command prompt type: **C:\> COPY A:\DOS\*.* C:\X12DOS** and press return. This will copy all the file required to the X12DOS directory.
10) At command prompt type: **C:\> CD \X12DOS** and press return to enter the directory.
11) At command prompt type: **C:\X12DOS> VSETUP –Z** and press return. This initializes all the 512 parameters to their default state.
12) If you are using a **PAL** video source, type: **C:\X12DOS> VSETUP –PAL** and press return. For NTSC you can skip this step.
13) At command prompt type: **C:\X12DOS> X12TSR** and press return. This step load the driver. Now the demo application can be run. Video should now appear on the monitor.
14) At command prompt type: **C:\X12DOS> RECORD TEST.MPG 300 50** and press return. This will record an MPEG video clip approximately 300 frames (10 seconds for NTSC, 12 soconds for PAL) long. Note: The 50 is a 'cleanup' parameter and delays the termination in order to allow the video queue to be processed. This parameter should be determined empirically for best results. If at playback the clip is shorter than expected increase this 'cleanup' number. TEST.MPG is the file name.
15) command prompt type: **C:\X12DOS> PLAYBACK TEST.MPG** and press return. This will playback the recorded video clip.
16) For different video parameters run **VSETUP.EXE** again and proceed to step 14 for testing.

RECORD.EXE and **PLAYBACK.EXE** are demo programs (source code included) that can be used to try out the 512. Typing RECORD or PLAYBACK at the command line without any parameters will give you a list of available parameters and options.

Appendix B: Technical support

For technical support contact Sensoray Company Inc.

Tel:	(503) 684-8073
Fax:	(503) 684-8164
e-mail: support@sensoray.com
WWW: www.sensoray.com