

Matrox Intellicam

version 2.07

User Guide

Manual no. 10521-301-0302

July 7, 2000

Matrox® is a registered trademark of Matrox Electronic Systems Ltd.

Microsoft®, MS-DOS®, Windows®, and Windows NT® are registered trademarks of Microsoft Corporation.

Intel®, Pentium®, and Pentium II® are registered trademarks of Intel Corporation.

Texas Instruments is a trademark of Texas Instruments Incorporated.

RAMDAC™ is a trademark of Booktree.

All other nationally and internationally recognized trademarks and tradenames are hereby acknowledged.

© Copyright Matrox Electronic Systems Ltd., 2000. All rights reserved.

Disclaimer: Matrox Electronic Systems Ltd. reserves the right to make changes in specifications at any time and without notice. The information provided by this document is believed to be accurate and reliable. However, no responsibility is assumed by Matrox Electronic Systems Ltd. for its use; nor for any infringements of patents or other rights of third parties resulting from its use. No license is granted under any patents or patent rights of Matrox Electronic Systems Ltd.

PRINTED IN CANADA

Contents

Chapter 1: Introduction 9

Welcome to Matrox Intellicam	10
What you need to run Matrox Intellicam.	11
Installation	12

Chapter 2: Overview 15

Overview	16
The tool button bar.	18
The menu bar	20
The status bars.	21
Window management.	22
Multiple tabs.	22
On-line help	22
Illustrations and examples	22
Using Intellicam	23
Choosing a target system.	23
Creating a DCF document for your camera	24
Creating an image document.	26
Grabbing from your camera.	27
Experimenting with the digitizer controls	27
Using the MIL Interpreter.	28

Chapter 3: How to grab using a standard camera . . 31

Grabbing from a standard camera and adjusting the grab parameters	32
Choosing a system	33
Choosing a digitizer configuration format. . . .	34
Creating an image document	35
Grabbing from your camera	36
Fine-tuning your DCF	36
Using the Digitizer Control settings	38

Chapter 4: How to grab using a non-standard camera 41

Interfacing a non-standard camera	42
Concrete examples	43
Interfacing a non-standard frame scan camera.	43
Interfacing a line scan camera	59

Chapter 5: The Matrox Imaging Library Interpreter . . 63

What is the Matrox Imaging Library Interpreter?.	64
Getting started with MILINTER	65
MILINTER overview	65
Invoking MILINTER	65
MILINTER tool bar buttons.	66
Using MILINTER	67

Intellicam macros	69
Opened image documents	69
Allocated Intellicam system objects.....	70
Allocated Intellicam application object	70
Matrox Imaging Library (MIL) examples	71
Setting up command lists.....	72
Notational conventions.....	72
Matrox Imaging Library (MIL) standard types and macros	73
MIL standard variable types	73
MIL macros.....	75
Help	77
Command definition	78
Parameter definition	80
Constant	80
Numerical expression.....	81
Editing	85
History	86
Manipulation of variables	87
Manipulation of numbers.....	87
Manipulation of strings	88
Manipulation of arrays.....	90
MILINTER macro control commands	93
Command lists	94
Editing command lists	96

Advanced features	97
Conditional operation	97
Loop	98
Timer	100
I/O - keyboard and Host display.	101
File handling	102
Miscellaneous commands	103
Os shell	105
Error messages.	106
Interpreter Syntax	111

Appendix A: Interfacing a camera 113

Overview.	114
Video formats	115
Standard video formats	115
Non-standard video formats.	115
Standard analog video signal	116
Interlaced and non-interlaced signals	116
Synchronization	117
Back and front porch	118
Active video	118
Amplitude and reference levels	118
Blanking periods.	118
Pixel clock	121
Color timing.	123
Non-standard video formats	125
Negative-going video.	125
Digital video	125

Camera modes of operation.....	128
Frame scan cameras.....	128
Line scan cameras	136
Summary of camera modes	139
<hr/>	
<i>Appendix B: Glossary.....</i>	<i>141</i>
Glossary of terms.....	142
<hr/>	
<i>Appendix C: Video Specification Form</i>	<i>155</i>
Providing us with your video specifications....	156
Video Specification Form.....	157
<hr/>	
<i>Appendix D: Troubleshooting</i>	<i>163</i>
Troubleshooting.....	164
Keying.....	164
Grabbing	164
Loading DCF files created with Intellicam 1.0	165
<hr/>	
<i>Index</i>	
<hr/>	
<i>Product Support</i>	

Chapter 1: Introduction

This chapter briefly describes the features of the Matrox Intellicam configuration software and how to install it.

Welcome to Matrox Intellicam

Matrox Intellicam 2.0 is a high-level, 32-bit Windows-based program that provides fast camera interfacing and interactive access to all the grab features and functionality of your Matrox digitizer.

It allows you to:

- Interface with various camera types.
- Interactively experiment with your digitizer.
- Use and experiment with the various Matrox Imaging Library (MIL) functions, using the MIL Interpreter.

When you are using a digitizer that supports non-standard acquisition, you can use Intellicam to create and/or modify digitizer configuration format (DCF) files. These files can then be used to interface to any camera that is supported by your Matrox digitizer.

Intellicam also includes interactive grab and control features.

What you need to run Matrox Intellicam

To run Intellicam, you need the following:

- A PC with a Pentium (or better) processor, at least 8 MBytes if you are using Windows 95 or 16 MBytes of memory if you are using Windows NT, and a hard disk with approximately 20 MBytes of free space or approximately 25 MBytes if you are also installing the Matrox Imaging Library (MIL or MIL-Lite).
- An installed version of Windows 95 or Windows NT.
- A display adapter, such as a SVGA board. This is optional if the digitizer has an integrated display adapter.
- A Windows-supported mouse.
- An installed digitizer (optional).

Installation

Intellicam comes with MIL or MIL-Lite. Simply install MIL (or MIL-Lite) and Matrox Intellicam will automatically be installed along with it. If you want to install only Intellicam, choose the appropriate option during the MIL installation procedure .

*Installing MIL
(or MIL-Lite)*

If you are running under Windows 95:

1. Start Windows.
2. Insert the MIL (or MIL-Lite) CD-ROM in the appropriate drive.
3. Click on the **Start** button and select **Run** from the **Start** menu.
4. Type the following command, using the appropriate drive letter, in the **Command Line** box:

d:\setup

If you are running under Windows NT:

1. Start Windows.
2. Insert the MIL (or MIL-LITE) CD-ROM in the appropriate drive.
3. In Program Manager, select **Run** from the **File** menu. If you're working with Windows NT 4.0, click on the **Start** button and select **Run** from the **Start** menu.
4. Type the following command, using the appropriate drive letter, in the **Command Line** box:

d:\setup

5. Click on the **OK** button.

While installing the software, you will be asked a number of questions, for example:

- The drive and directory in which to install MIL (or MIL-Lite).
- The compiler you are using, and where it is located on your disk.
- Whether or not you want support for Visual Basic.
- The type of Matrox hardware that is installed in your system.
- Some board-specific questions.



Once installation is complete, a MIL group, which includes the Intellicam application icon, is created. You can then create a shortcut to Intellicam on your desktop.

Creating a shortcut for Intellicam

To create a shortcut, select **Explore** after clicking the right mouse button on the **Start** button. From the **Exploring** dialog box, select the MIL\INTELCAM directory. Select the *intelcam.exe* file and drag it to your desktop. Press the **Ctrl** key and drop the file on your desktop. Pressing the **Ctrl** key ensures that you are placing a copy of the *intelcam.exe* file on your desktop and not the *intelcam.exe* file itself.

The Intellicam shortcut icon should appear.

Chapter 2: Overview

This chapter describes the Matrox Intellicam basics required to get started.

Overview

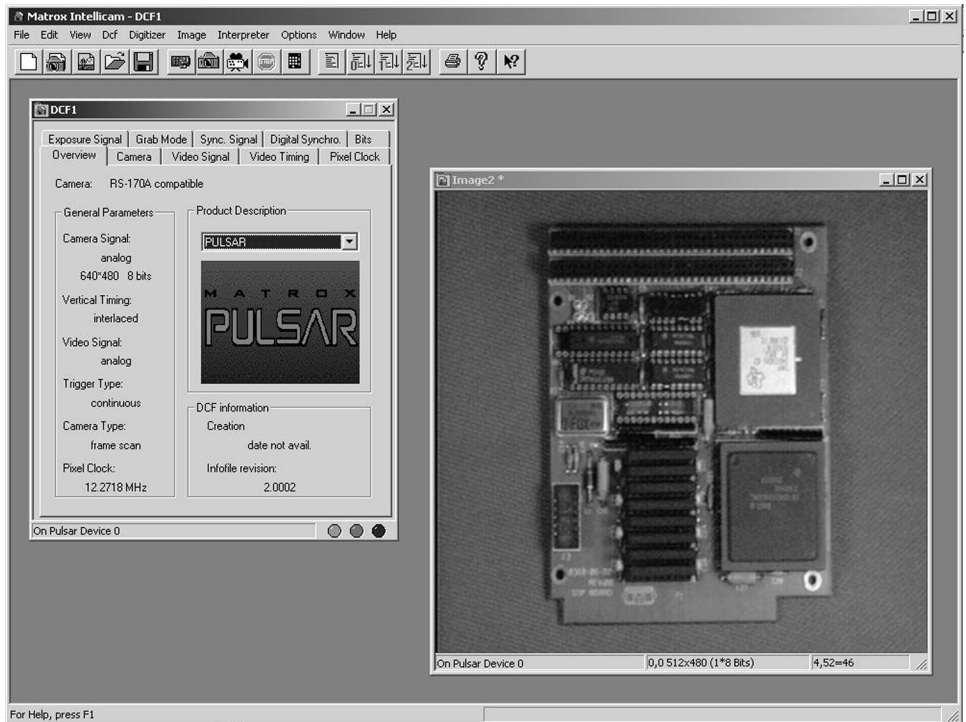
Matrox Intellicam 2.0 is a high-level, 32-bit Windows-based program that provides fast camera interfacing and interactive access to all the grab features and functionality of your Matrox digitizer.

Matrox Intellicam is Multiple Document Interface (MDI) compliant and follows the general conventions of the Windows Graphical User Interface (GUI). You can refer to your Windows documentation for specific explanations of the windows conventions, such as drop-down menus, icons, and dialog boxes. This section describes the following application-window elements:

- The tool button bar
- The menu bar
- The status bars

It also covers window management, multiple tabs, and on-line help, as well as illustrations and examples.

Once installation is complete (see Chapter 1), you can load Intellicam by clicking on its icon. The first screen you will see is the Matrox Intellicam opening screen, which contains a menu bar, a tool button bar, an application workspace, and a status bar. (In the illustration below, a DCF document and an image window are also shown).



Upon loading, you will also be presented with the **Tip of the Day** window, which provides you with useful tips on Intellicam. You can review all the tips by clicking on the **Next Tip** button. You have the option of continuing to view such tips every time you load Intellicam or discontinuing them by disabling the **Show Tips at StartUp** check box. The **Tips** can also be accessed from the **Help** menu.

The tool button bar

By default, the tool button bar is located underneath the menu bar (described in the next section). The button bar allows you to quickly access the most common commands in Intellicam.



1. **New:** Create a document (DCF or image).
2. **New DCF:** Create a new DCF document.
3. **New Image:** Create a new image document.
4. **Open:** Open an existing document file (DCF or image).
5. **Save:** Save the active document (DCF or image).
6. **System Selection:** Display a list of available systems and change the active system.
7. **Single Grab:** Single camera grab into the active image buffer, using the active DCF.
8. **Continuous Grab:** Continuous camera grab into the active image buffer, using the active DCF.
9. **Halt Grab:** Halt a continuous grab.
10. **Digitizer Control:** Display or hide the digitizer control window.
11. **MIL Interpreter:** Display or hide the interpreter command window.
12. **Interpreter Command 0:** Call a user-defined interpreter command.
13. **Interpreter Command 1:** Call a user-defined interpreter command.
14. **Interpreter Command 2:** Call a user-defined interpreter command.
15. **Print:** Print the active document.
16. **Help Contents:** Display the contents of the on-line help.
17. **Help:** Obtain context-sensitive help for various buttons, menus and windows.

*Positioning the tool
button bar*

The tool button bar is dockable to the top, bottom, left, or right edge of the Intellicam application workspace. Otherwise, it appears as a window that you can position wherever you like on the Windows desktop. To move the button bar, place your cursor anywhere on the bar, except over the buttons, and drag it to the required position.

The menu bar

The menu bar is located just above the tool button bar. It contains drop-down menus that list various commands. Some commands are available only when the appropriate window is active. Commands followed by ellipses (...) indicate that a dialog box opens upon selecting the command.

File <u>N</u> ew Ctrl+N <u>O</u> pen ... Ctrl+O C <u>l</u> ose <u>S</u> ave Ctrl+S S <u>a</u> ve A <u>s</u> ... <hr/> <u>P</u> rint Ctrl+P P <u>r</u> int P <u>r</u> ev <u>i</u> ew P <u>r</u> int S <u>e</u> tup ... <hr/> <u>1</u> C:\MIL\pulsar\dcf\Rs170 <hr/> <u>E</u> x <u>i</u> t	View A <u>s</u> Editor A <u>s</u> Report A <u>s</u> Text <hr/> <input checked="" type="checkbox"/> T <u>o</u> olbar <input checked="" type="checkbox"/> S <u>t</u> atus Bar <hr/> <input checked="" type="checkbox"/> C <u>h</u> ild A <u>r</u> ea (When in a DCF Window)	Interpreter C <u>o</u> mm <u>a</u> nd W <u>i</u> nd <u>o</u> w P <u>r</u> ef <u>e</u> rences ...
Dcf <u>V</u> alidate DCF <u>C</u> ompute Registers <hr/> S <u>h</u> ow E <u>r</u> ror R <u>e</u> port S <u>h</u> ow A <u>u</u> to-C <u>o</u> rr <u>e</u> ct R <u>e</u> port <hr/> P <u>r</u> ef <u>e</u> rences ...	View A <u>s</u> Editor A <u>s</u> Report A <u>s</u> Text <hr/> <input checked="" type="checkbox"/> T <u>o</u> olbar <input checked="" type="checkbox"/> S <u>t</u> atus Bar <hr/> <input checked="" type="checkbox"/> C <u>h</u> ild A <u>r</u> ea (When in an Image Window)	Options S <u>y</u> stem S <u>e</u> lection ... P <u>r</u> ef <u>e</u> rences ...
D<u>i</u>gitizer S <u>i</u> ngle G <u>r</u> ab C <u>o</u> ntinu <u>o</u> us G <u>r</u> ab <hr/> C <u>o</u> n <u>t</u> rol	I<u>m</u>age C <u>h</u> ild A <u>r</u> ea ... <u>I</u> mport ... <u>R</u> eload <hr/> P <u>r</u> ef <u>e</u> rences ...	Window <u>N</u> ew Window C <u>a</u> scade <u>T</u> ile <u>A</u> rrange I <u>c</u> ons <u>C</u> lose All <hr/> <input checked="" type="checkbox"/> 1 Image1
		H<u>e</u>lp H <u>e</u> lp T <u>o</u> p <u>i</u> c <u>s</u> H <u>o</u> w to U <u>s</u> e H <u>e</u> lp <hr/> T <u>i</u> p of the D <u>a</u> y <hr/> A <u>b</u> out I <u>n</u> tellicam ...

Context menus

In addition to the above-mentioned menus, Intellicam also includes context menus. Context menus are available for all document windows, such as the DCF and image document windows, as well as the MILINTER command window. To access a context menu, move the cursor over the appropriate window and click the right mouse button. The context menu for that window is displayed. From this window, you can select the required command.

The status bars

Status bars are located at the bottom of various windows:

*DCF document
window status bar*

The status bar at the bottom of the DCF document window displays information about the system to which it is associated, and a "traffic light" type signal. When the DCF is configured properly, the "green light" is active. If there is an error in the current tab that can be corrected by adjusting other values in the same tab, the "yellow light" is active. If there is an error that can be corrected by adjusting values in another tab(s), the "red light" is active.

*Image document
window status bar*

The status bar at the bottom of the image document window displays the owner system (that is, the system for which the image is allocated), the coordinates of the child area (relative to the (0,0) coordinates of the image window), the horizontal and vertical dimensions of the child area, the pixel depth (that is, the number of bands x the number of bits per band), the coordinates of the pixel under the cursor (relative to the (0,0) coordinates of the whole image) and its value.

*Application workspace
status bar*

The status bar at the bottom of the application workspace displays different information messages. When you place the cursor over a tool button, the status bar displays a short description of the functionality of that tool button. When an error occurs, the status bar flashes the MIL error message that corresponds to that error.

Window management

The windows (and icons) in Intellicam can be managed in the same way as those in other Windows applications. They can be maximized, minimized, dragged, resized, rearranged (tiled or cascaded), opened, and closed. Icons can be dragged and rearranged.

Multiple tabs

Several of Intellicam's dialog boxes are actually multiple dialog boxes, that is, different dialog boxes overlaid one on top of the other. The main dialog box is referred to as a dialog box, while its internal dialog boxes are referred to as tabs. The tab name appears at the top of the tab. To access one of these tabs, click on the appropriate tab.

On-line help

Note that this manual is designed to guide you smoothly through the general operations of Intellicam, whereas the on-line help included in Intellicam is designed to provide more task-specific assistance. For example, the on-line help includes descriptions of all the Intellicam commands.

Illustrations and examples

All the illustrations in this manual reflect the Windows 95 and Windows NT 4.0 interface. The Windows NT 3.51 interface is slightly different, but the functionality of the Intellicam software is identical.

The target system used in all the examples is a **Matrox Pulsar**.

Using Intellicam

Choosing a target system

You must ensure that Intellicam is configured for the system you are using. In Intellicam, a system generally refers to a board with video acquisition capabilities that is installed on your computer. In addition to the actual systems available, you can choose a "virtual system".

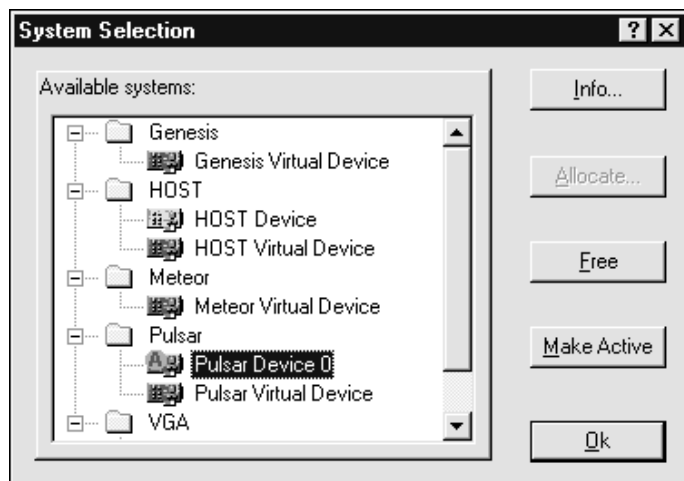
Virtual systems

Choose a virtual system if you do not actually have a particular board installed, but would like to set up a DCF document file for use with that board in the future. Note that, when you allocate an image on a virtual system, it is allocated on the VGA system.

The first time you load Intellicam, the **System Selection** window appears. From this window, you can choose the required system. At any other time, choose a system as follows:



1. Click on the **System Selection** button. The **System Selection** dialog box opens.



2. Choose your target system, for example, **Pulsar Device 0**, and double click on it. This will make the system you have chosen the active system.

Device 0 refers to the fact that the active board is either the only board installed on your computer, or it is the first of several boards. Moreover, it indicates a physical board as opposed to a virtual one.

3. You will be prompted to confirm the setting you have chosen. Click on **OK**, thereby allocating this system and selecting it as the active system where captured and stored images and DCFs will be associated.

When you double click on a chosen system, it becomes the default system and the icon next to your selection changes to an **A** (for Active System).

4. Click on **OK** to confirm your choice.

Creating a DCF document for your camera

With Intellicam, you can use two different types of documents: digitizer configuration format (DCF) files and images (TIFF or MIM). These can be new documents or existing documents that are stored on disk. DCF documents are discussed below, while image documents are discussed in the next section.

A DCF is a document that specifies your video timings and format. It contains the register-level information that must be downloaded into the digitizer section of your target system (that is, the system you will be using with your application) before you can grab with it, using your particular camera.

To interface a camera, you must specify the video timings and format of that camera. In essence, specifying a DCF defines the initial digitizer configuration to match your particular camera type. This configuration specifies the initial state of the digitizer and the video signal format it will be configured to receive.

If you are using a standard camera (such as RS-170 or CCIR), a digitizer configuration format has already been defined for your camera. If you are using a non-standard camera, you should check to see if a DCF has already been created for that particular camera. If not, you will either have to modify an existing DCF or create a new one.

Pre-defined DCF document files



Intellicam allows you to load pre-defined DCF files that you can use to interface your particular camera. In general, these files are located in the `\DCF` directory of your target system. For example, you can view the details of the various formats for the Matrox Pulsar in the `dcf_list.txt` file located in the `\MIL\PULSAR\DCF` directory.

To open an existing DCF, click on the **Open** button and locate the DCF document file you want to open.



❖ Note that using the **Save** button when you have an existing DCF open will overwrite the original file. If you want to keep the original file, use **File - Save As** to save your new file, giving it a different filename.

You can also create your own DCF, but it is easier to start from an existing DCF and simply change the fields that do not conform to the configuration of your particular camera.

Creating your own DCF



To create a DCF document, click on the **New DCF** button. Either select an initial digitizer format from the presented dialog box or click on the **Browse** button to start from an existing DCF file.

The document window that is created will not have the same name as the initial format you have chosen; rather, it will have a temporary name, for example, DCF1. You can choose a filename when you save your DCF, using the **Save** button.

Creating an image document

In addition to digitizer configuration format (DCF) document files, you can also use images (TIFF or MIM) with Intellicam. An image document is a memory buffer associated with your target system. It is used to store grabbed or loaded images.



To create an image document, click on the **New Image** button. The **New Image** dialog box opens, presenting you with various options from which you can choose the size, type and attributes of your image. If you want to open more than one new image, click on **Apply** to open each new image. When you are finished, click on **OK** to continue.

If a DCF is already opened in Intellicam, you can create an image document that has the dimensions specified in the current DCF (displayed in the **Overview** tab, under **Camera signal**) by enabling the **Fill with current digitizer sizes** check box in the **New Image** dialog box.

You may want to define a child area in your image document. For instance, you may want to work only a specific region of interest within the image itself. To define such an area, double click at the desired position in the image document and drag the child area to the desired size.

Grabbing from your camera

*Grabbing a
single image*



With Intellicam, you can either grab a single image or grab continuously (live grab).

To grab a single image into an image document using the active DCF, click on the **Single Grab** button. You can now work on your image as required.

*Grabbing
continuously*



To grab continuously into an image document using the active DCF, click on the **Continuous Grab** button and your images will be grabbed live.



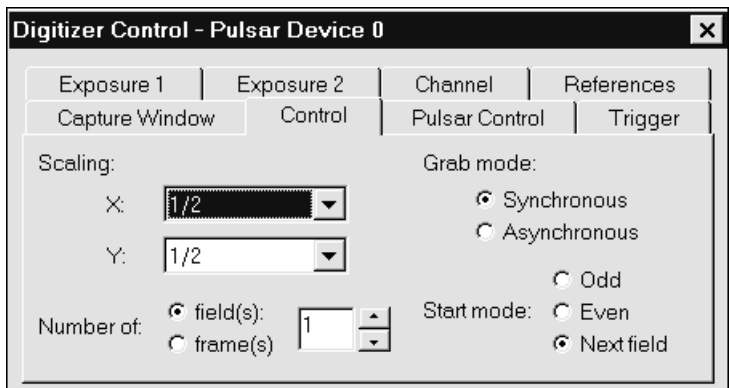
To halt a grab, simply click the **Halt Grab** button.

Experimenting with the digitizer controls

You can experiment interactively with the various dynamic controls of your digitizer. Simply start a continuous grab (click on the **Continuous Grab** button), and experiment with the various controls provided in the **Digitizer Control** dialog box.



To open the **Digitizer Control** dialog box, click on the **Digitizer Control** button.



The **Digitizer Control** dialog box is unique for each board, and provides all the digitizer control options available for your particular board. Choose among the various tabs to access the digitizer controls. For example, select the **Control** tab to set **Scaling X** and **Scaling Y** to 1/2. The live grab is then performed at half the normal size.

It is important to note that each control in the various tabs corresponds to a digitizer setting accessible through a MIL function (for example, *MdigControl()*, *MdigReference()*, *MdigChannel()*). Given this fact, you can find the exact value you require for each setting and see its effect before you write your application program.

As you choose various controls, you can immediately see the results of your changes on the image you are grabbing. This allows you to become familiar with the capabilities of your board when it is initialized with a particular DCF.

❖ Note that you cannot save the settings you change in the **Digitizer Control** dialog box since these are dynamic controls applied "on top of" the initial DCF state. However, if you continue to use the same DCF with the same digitizer during the same session, your settings will be "saved" for the duration of that session.

Using the MIL Interpreter

Intellicam includes a command-line interpreter (called MILINTER) that gives you access to all the functions provided with MIL (or MIL-Lite). MILINTER lets you experiment with the various functions without having to recompile every time a change is made. Command lists (or scripts) can be created and run to perform a sequence of MIL commands.

Since the MILINTER syntax is similar to C language grammar, experimenting with the MIL functions is easy for C programmers. This similarity also means that you can develop C programs based on command lists.

MILINTER also comes with various MIL examples, presented as command lists, to help you learn the MIL functions without having to set up a programming environment.



Start MILINTER by clicking on the **MIL Interpreter** button (or choosing **Interpreter - Command Window** from the menu bar). The MILINTER command-line window opens. Here, you can enter the commands you need. The command-line window also displays a list of pre-defined buffer names for the active image

Images can be accessed using their names. For example, entering *MbufClear Image1 128* will clear the contents of the "Image1" document to the value 128 (gray). Alternatively, you can use the ACTIM and ACTCH macros. For example, entering *MbufClear ActIM 128* will clear the contents of the active image document, while *MbufClear ActCH 128* will clear only the contents of a defined child area (sub-region).

The *ACTIM* and *ACTCH* macros always point to the active image document and to its associated active child area, respectively. As in any image document, a child area can be defined by double clicking at the desired position in the image document and dragging the child area to the desired size.

User-defined buttons

Intellicam lets you define the functionality of three interpreter buttons on the button bar, thereby enabling you to execute frequently used commands quickly and efficiently. In other words, each of these three buttons can be associated with a MILINTER command that executes when you click on the button. You can define the functionality of these buttons using the **Interpreter - Preferences** from the menu bar, and setting the **Category** to **Command Buttons**. By default, the Interpreter Command 0 button has the "Clear the active image to 128 (*MbufClear ActIM 128*)" command associated with it. If you load or grab an image and click on the **Interpreter Command 0** button, the active image will be cleared to gray (128)

From **Interpreter - Preferences**, you can also obtain **File path information** and change the location of your **Startup file**.

For more details on MILINTER, see Chapter 5.

Chapter 3: How to grab using a standard camera

This chapter gives you a step-by-step description of how to grab using a standard camera, and how to adjust the grab parameters.

Grabbing from a standard camera and adjusting the grab parameters

Following is a step-by-step description of how to choose a system and a digitizer configuration format to acquire your images, using Intellicam. This chapter also explains how to use the digitizer control settings.

Before starting, make sure that your digitizer is correctly installed and set up, and that your camera is properly connected to it.

Load Intellicam by clicking on its icon. Then,

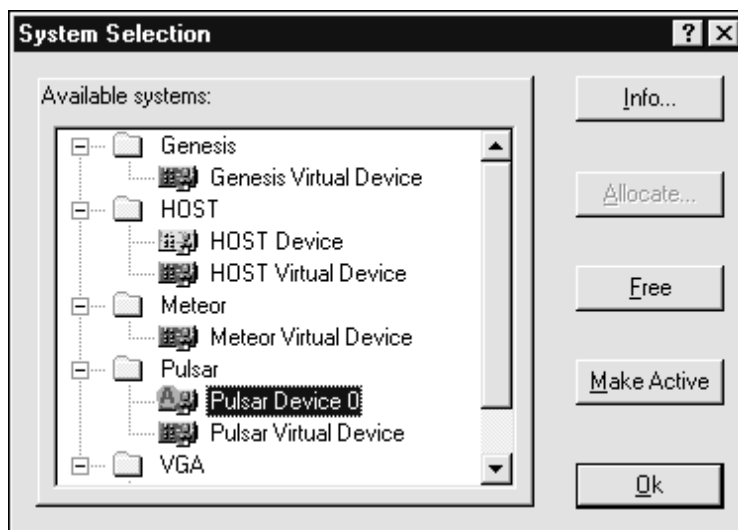
1. Choose a system.
2. Choose a digitizer configuration format.
3. Create an image document.
4. Continuously grab from your camera.
5. Fine-tune your DCF.

Choosing a system

To choose a target system (board):



1. Click on the **System Selection** button or choose **System Selection** from the **Options** menu. The **System Selection** dialog box opens. From this dialog box, you can allocate the system you will be using to acquire your images.



As mentioned previously, for the purposes of the examples given here, this system is a **Matrox Pulsar** frame grabber.

2. Double click on your target system (for example, **Pulsar Device 0**) to allocate the system and make it active.
3. Click on **OK** to close the **System Selection** dialog box.

Choosing a digitizer configuration format

To be able to grab from your camera, you must specify its video format by opening an existing digitizer configuration format (DCF) document or by creating one.

Opening an existing DCF document



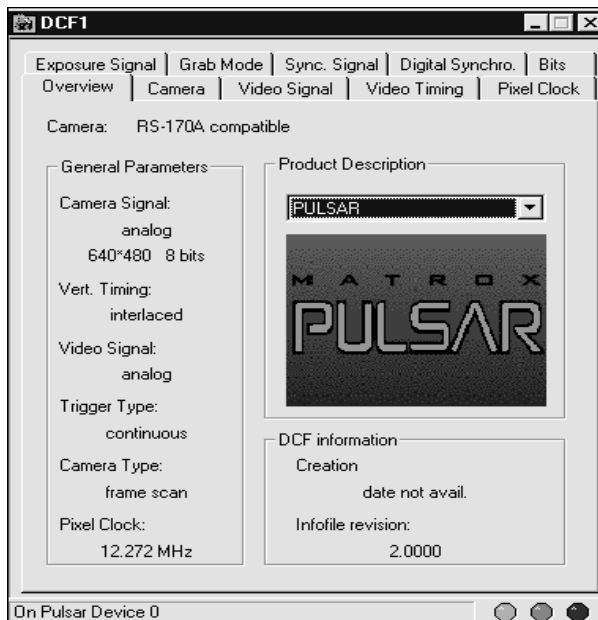
Creating a DCF document



To open an existing DCF document, click on the **Open** button and select the appropriate DCF document for your target camera. For example, the pre-defined DCFs for the **Matrox Pulsar** are located in the `\MIL\PULSAR\DCF` directory.

To create a DCF document, click the **New DCF** button or select **New** from the **File** menu. Choose an initial format for the DCF (RS-170 or CCIR), and then click on **OK**.

The **DCF** document dialog box opens with the **Overview** tab selected, as shown.



The visual presentation of this tab changes with the particular board you are using, with the biggest difference being between a board that supports only standard cameras and one that supports non-standard cameras and configurable DCFs.

In our example, since the **Matrox Pulsar** supports configurable DCFs, the **DCF** document dialog box opens with 10 tabs: Overview, Camera, Video Signal, Video Timing, Pixel Clock, Exposure Signal, Grab Mode, Sync. Signal, Digital Synchro. and Bits.



Details on a specific tab can be found by clicking on the **Help** button and then on the required tab. Or, make sure that the required tab is selected and press **F1**.

Creating an image document

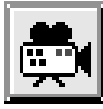


To create an image document, click on the **New Image** button, or select **New** from the **File** menu. This opens the **New Image** dialog box where you can select the size, type and attributes for the new image document.

By default, the **Fill with current digitizer sizes** check box is enabled if a DCF is opened. In such a case, the various image characteristics settings are not available since Intellicam automatically chooses those of the current DCF. Should you want to change any of the settings, (for example, if you are grabbing an image that you will want to enlarge later), simply remove the check. If no DCF is opened, this option is not available.

Click on **OK**. An empty image document window opens.

Grabbing from your camera

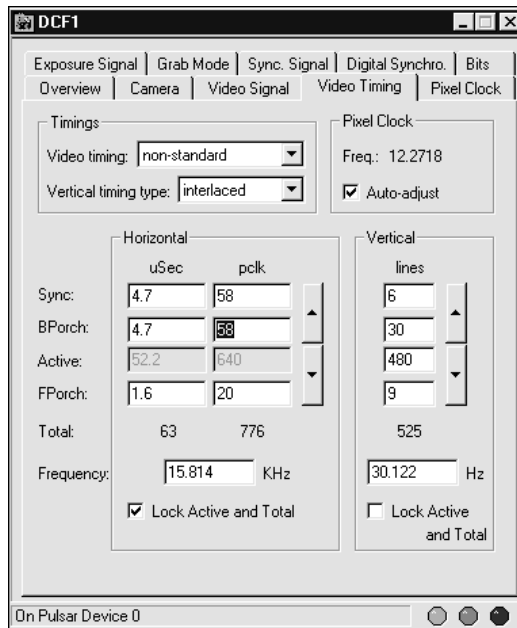


To grab continuously into this image document, make sure it is active, and then click on the **Continuous Grab** button.

Fine-tuning your DCF

If your digitizer supports configurable DCFs, you can fine-tune your video parameters to best suit your requirements. For the purposes of this example, let's assume that the active video is not properly centered because the back porch specified in the DCF is a little too narrow. To adjust the alignment, you will have to make the back porch a little wider.

- ❖ The "back porch" is the blank region at the left of the grabbed image. This blank region may be visible if the back porch specified in the DCF is not wide enough.
1. To adjust the back porch, thereby centering your image, click on the **Video Timing** tab of the DCF document while grabbing.



2. Enable the **Lock Active and Total** check box in the **Horizontal** section to keep the original total horizontal timings and have Intellicam automatically calculate the front porch value.
3. To change the setting for your back porch, type in the new value in the **Horizontal BPorch** edit field and press **Enter**. Otherwise, click on the value you want to change, and then click on the up or down arrows located on the right side of the window. When you click on the up arrow, the black stripe on the left of the grabbed image will disappear as the image is centered.
 - ❖ You can specify **Horizontal** timings in either microseconds (μ s) or pixel clock cycles.

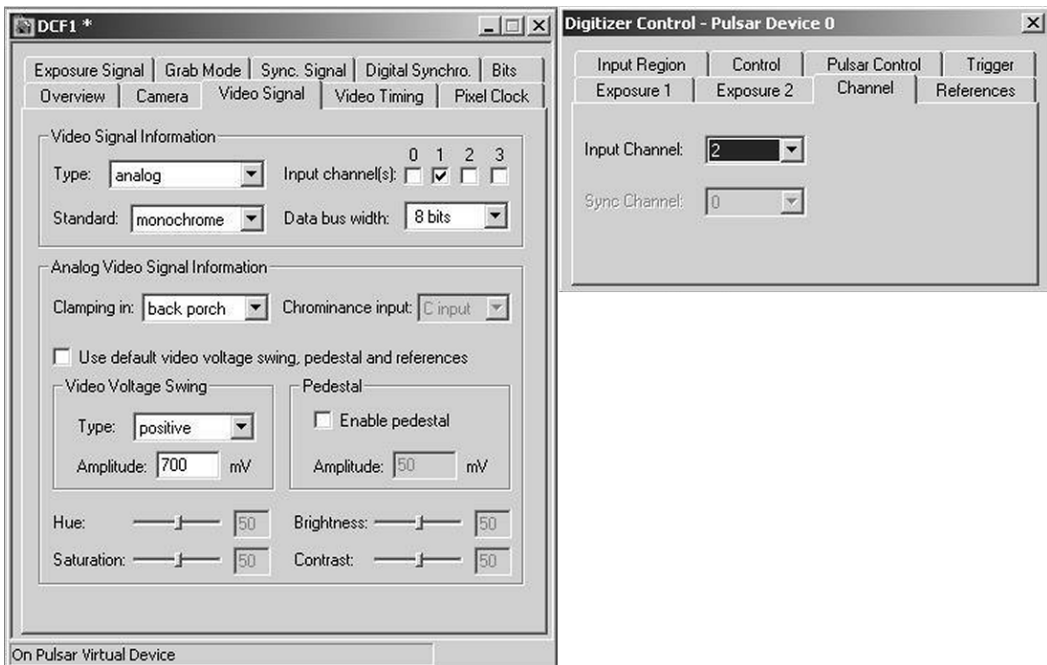
You will immediately see the result of your changes on the image you are grabbing. Continue to make adjustments until you are satisfied with the results.

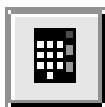
4. You can save your new DCF document by clicking on the **Save** button or by selecting **Save** from the **File** menu.

Using the Digitizer Control settings

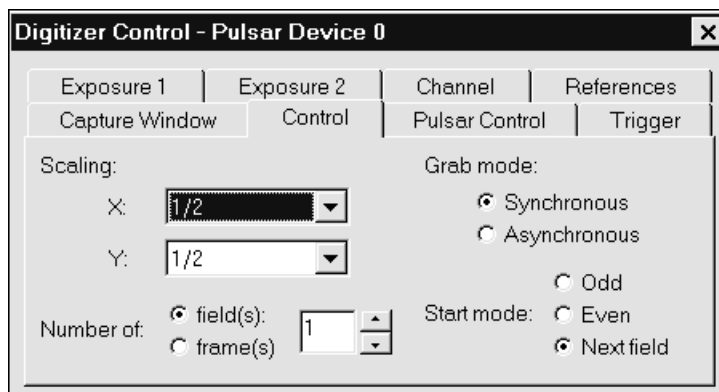
As you are creating or modifying your DCF, you are creating the "initial state" of the digitizer after its allocation. Some of the initial acquisition parameters, called digitizer controls, that you create can, however, be modified and/or adjusted. This can be done either dynamically in the **Digitizer Control** dialog box, or using the board control library, generally, MIL or MIL-Lite.

Any adjustments made using the board control library or the **Digitizer Control** dialog box will not change the contents of the DCF since the digitizer controls are simply applied "on top of" the DCF initial state. In other words, if you specify **Channel 1** as the default channel in your DCF (using the **the Input channel** field in the **Video Signal** tab) and set the **Input channel** field in the **Channel** tab of the **Digitizer Control** dialog box at 2, your grab will be done using Channel 2. However, when you reload the DCF, Channel 1 will still be the default channel.





To dynamically modify the state of the digitizer, click on the **Digitizer Control** button or select **Control** from the **Digitizer** menu. The **Digitizer Control** dialog box opens. You can, for example, use this dialog box to set the input channel, vary the black and/or white reference levels, or change the grab scale factors.



For example, the initial resolution setting for a standard RS-170 camera is 640 x 480. If you want to grab at, say, half that size, you can set the **Scaling X** and **Scaling Y** fields of the **Control** tab to 1/2. All subsequent grabbing will be done using that scale factor.

Chapter 4: How to grab using a non-standard camera

This chapter gives you a step-by-step description of how to grab using a non-standard frame scan camera and a line scan camera.

Interfacing a non-standard camera

If your target digitizer supports non-standard camera acquisition and downloadable digitizer configuration format (DCF) files, you can create and/or modify a DCF to interface your particular camera.

Once you have tested your hardware and software by grabbing with a standard camera (see Chapter 3), you can connect the non-standard camera you want to interface.

There are two major types of non-standard cameras:

- frame scan cameras
- line scan cameras

In this chapter, we will give you concrete examples on how to interface each of these cameras.

Before going on, you should become as familiar as possible with the camera you are using. Read the manufacturer's documentation carefully. For additional help, fill out the Video Specification Form provided in this manual, or send it to your manufacturer and ask him to fill it out for you.

When interfacing a non-standard camera, it is a good idea to start with an existing DCF that has a configuration that is similar to that of your camera. To choose the most appropriate DCF, look in the `\DCF` directory of your target board and view the headers of the various DCF files. For example, for the **Matrox Pulsar**, look in the `\MIL\PULSAR\DCF` directory. Here, the `dcf_list.txt` file contains a summary of all the DCFs available for the **Matrox Pulsar**.

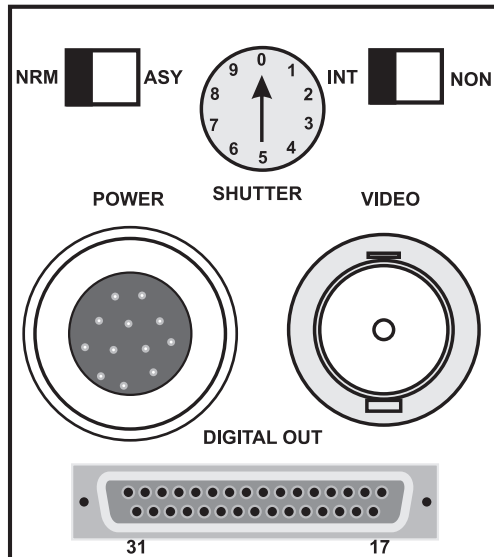
Concrete examples

Following are examples that give you a step-by-step description of how to interface a non-standard frame scan camera and a line scan camera. In other words, these examples show you how to create DCF files to interface the different modes of such cameras, and how to control the various options these feature.

Interfacing a non-standard frame scan camera

In this first example, we will interface a non-standard frame scan camera in various modes. Our target camera is the fictitious FSCAM. The FSCAM is a very flexible camera that supports RS-170 interlaced (640 x 480 at 12.2 MHz @ 30 frames per second) or non-standard high-resolution non-interlaced mode (768 x 484 at 14.3 MHz @ 30 frames per second). The FSCAM can output analog RS-170 compatible data or digital RS-422 8-bit data. It can be used in continuous scan mode (normal) or in asynchronous reset trigger mode. It can also handle variable exposure times (shutter control) for the CCD. Our target digitizer is the **Matrox Pulsar**.

FSCAM BACK PANEL



Interlaced mode

First, let's create a DCF to use the camera in its standard RS-170 interlaced mode. To do this:

1. Using a standard analog cable, plug the analog output of the camera to the analog input on your Pulsar frame grabber, and set the switches to normal mode and interlaced (NRM+INT). Refer to the illustration of the FSCAM back panel.



2. Click on the **System Selection** button or choose **System Selection** from the **Options** menu. Then, choose **Pulsar Device 0** as your target system.

3. Create a new DCF initialized with the existing RS-170 DCF:



- a. Click on the **New DCF** button or choose **New** from the **File** menu. Then, make sure the **Digitizer Configuration Format** tab is active.

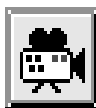
- b. Browse to find an existing DCF for a standard camera (RS-170). For example,
C:\MIL\PULSAR\DCF\RS170.DCF

- c. Click on **Apply**.



4. Open an image document window. To do so, click on the **New Image** button or choose **New** from the **File** menu. Then, make sure the **Image** tab is active. Since the **New documents** dialog box is already open, simply click on the **Image** tab.

Since you will be grabbing into this image document window, enable the **Fill with current digitizer sizes** check box. Then, click on **OK**.



5. Start a continuous grab by clicking on the **Continuous Grab** button. If necessary, adjust the DCF's video timings to suit your needs. To adjust the video timings, use the **Video Timing** tab in the **DCF** document window.

6. Update the camera information in the **Camera** tab. In the **Camera name** field, type *"FSCAM, RS-170A compatible"*. Also, update the **Comments** field, if necessary.



7. Save the new DCF in your target directory, under the name *FSCAMI.DCF*. To do so, click on the DCF to make it active, and then click on the **Save** button, or select **Save** from the **File** menu.

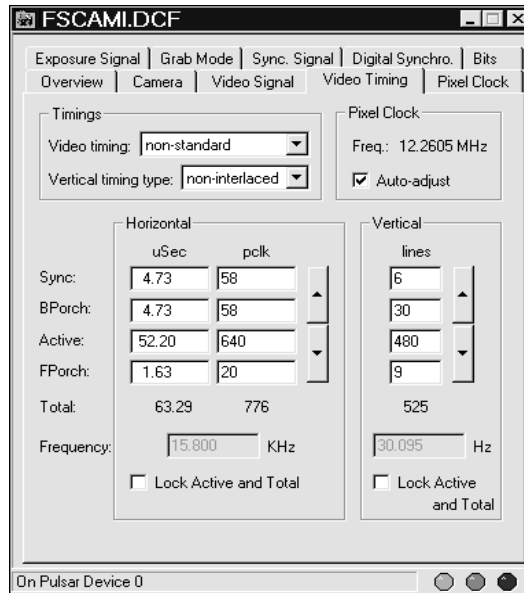


8. Stop the grab by clicking on the **Halt Grab** button.

Non-interlaced mode

Here, we will modify the DCF we just created for interlaced mode (*FSCAM.DCF*) to accept a non-interlaced RS-170-like signal:

1. Set the camera switch to non-interlaced (NON). Refer to the illustration of the FSCAM back panel.
2. Choose the **Video Timing** tab. From the **Vertical timing type** field, choose **non-interlaced**.



3. Click on the **Continuous Grab** button and adjust the DCF's video timings, if necessary, according to your requirements.
4. Update the camera information in the **Camera** tab. In the **Camera name** field, type "*FSCAM, RS-170, non-interlaced*". Update the **Comments** field, if necessary.
5. Save the new DCF under the name *FSCAMN.DCF*. To do so, click on the DCF with the right mouse button and select **Save As**, or use **File - Save As** from the menu bar.
6. Stop the grab by clicking on the **Halt Grab** button.

High resolution

Now, let's modify the DCF we just created for non-interlaced mode (*FSCAMN.DCF*) to take advantage of the camera's high-resolution feature (768 x484 at 14.3182 MHz):

1. Choose the **Pixel Clock** tab. In the **Pixel Clock** field, type in the new frequency value: 14.3182 MHz.
2. Choose the **Video Timing** tab.
 - a. Select the resolution of the camera by adjusting the horizontal timings of the DCF. To do so, select the **Active - pclk** field from the **Horizontal** section, and type in the new horizontal resolution: 768.

Adjust the back porch and the front porch to best match the camera's horizontal frequency (15.734 Hz). To do so, click on the **BPorch - pclk** field and use the up or down arrows to adjust the horizontal frequency. For 15.734 Hz, the back porch value should be 51 pclk.

Here, you are removing or adding pixels in the back porch until the horizontal frequency displayed in the **Frequency** field is at the required value.

- b. In the **Vertical** section, click on the **Active - lines** field and type in the value: 484.

Since our camera has a total of 525 lines (30 Hz) per frame, some of the blanking lines in the back porch and front porch have to be removed. To do so, click on the **Front Porch - lines** field and reduce the number of lines until the total lines equals 525 (The value in the **FPorch - lines** field should be 5).

The horizontal and vertical resolutions are now set. Now, we will interactively test and fine-tune our DCF.

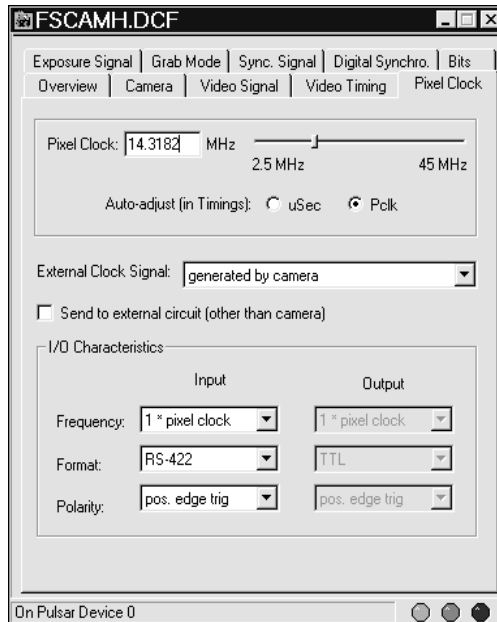
3. Close the previous image (if you no longer need it), and allocate a new image (of resolution 768 x484). To do so, click on the **New Image** button or select **New** from the **File** menu. Make sure the **Image** tab is active and then enable the **Fill with current digitizer sizes** check box.

4. Start a continuous grab by clicking on the **Continuous Grab** button.
5. Your image will not be properly centered, and a stripe of black pixels will be visible on both sides of the image. This means that the horizontal and vertical back and front porches have to be adjusted.
 - a. To center your image horizontally, adjust the **BPorch - pclk** and **FPorch - pclk** values in the **Horizontal** section. Use the up and down arrows to fine-tune the image you are grabbing. The final value for the back porch should be 64, and the final value for the front porch should be 24.
 - b. To center your image vertically, first enable the **Lock Active and Total** check box in the **Vertical** section. This "locks" the number of lines in the back and front porches to 525 as you are adjusting the blanking lines in the back and front porches. Then, click on the **BPorch - Lines** field and use the up and down arrows to fine-tune the image you are grabbing. The final value for the back porch should be 29, and the final value for the front porch should be 6.
6. Update the camera information in the **Camera** tab. In the **Camera name** field, type *"FSCAM, high resolution, non-interlaced"*. Also update the **Comments** field, if necessary.
7. Once you are satisfied with the quality of the grabbed image, save your new DCF under the name *FSCAMH.DCF*. To do so, click on the DCF with the right mouse button and select **Save As**, or use **File - Save As** from the menu bar.
8. Stop the grab by clicking on the **Halt Grab** button.

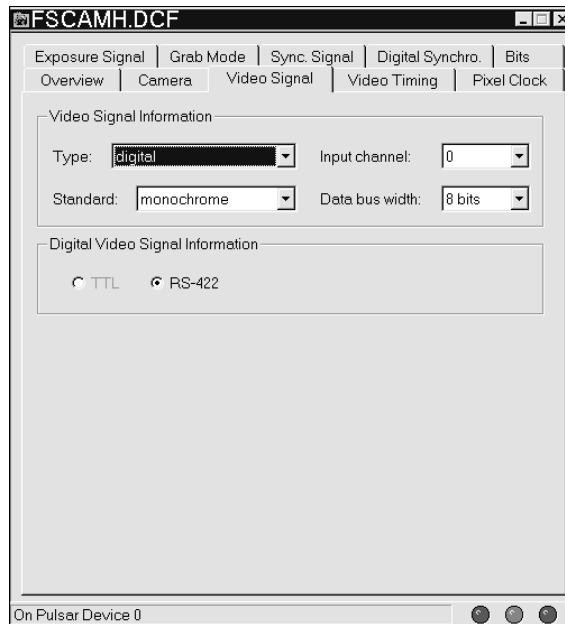
Digital data

In this example, we will modify the DCF we just created for high resolution (*FSCAMH.DCF*) to receive digital data and synchronizations:

1. Plug the camera's digital RS-422 output connector to the input of the Pulsar's RS-422 digital module, using a custom digital input cable. In general, the signals you will need to connect are: a pixel clock, hsync and vsync, and data. For technical information about this cable, see the *Technical Information* chapter in the hardware manual for your specific board (e.g., the *Matrox Pulsar Hardware and Installation Manual*).
2. Choose the **Pixel Clock** tab.



- a. From the **External Clock Signal** field, choose **generated by camera**.
- b. From the **I/O Characteristics** section, set the **Input - Format** field to **RS-422**. This indicates that the camera's RS-422 port generates the pixel clock.

3. Choose the **Video Signal** tab.

- a. From the **Type** field, choose **digital**. A pop-up window might display various error messages to indicate that certain parameters are no longer valid for a digital data DCF. The red light on the bottom right of the window will light up to indicate that the grab cannot be performed.

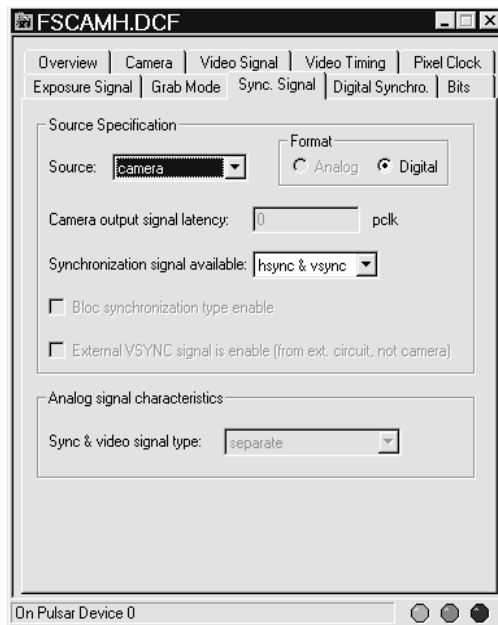
In this example, the following error message should appear:

Description: Analog sync. with current video type signal not supported by digitizer.

Corrections Needed: Select the Synchronization Signal menu to correct the error.

- b. Before moving to the **Sync. Signal** tab, complete the information on the **Video Signal** page by selecting **RS-422** in the **Digital Video Signal Information** field.

4. Choose the **Sync. Signal** tab.



- a. From the **Source** field, choose **camera**.
- b. From the **Format** field, choose **digital**, if it is not already selected.

A pop-up window might open, displaying the following error message:

Missing information in synchronization signals identification.

- c. From the **Synchronization signal available** field, choose **hsync & vsync**.
5. Select the **Digital Synchro.** tab. From the **HSync** and **VSyn** sections, enable the **Input Active** check boxes. Set the **HSync - Format** and **VSyn - Format** fields to **RS-422**, since we are grabbing from the RS-422 module of the **Matrox Pulsar**. Also, set the **HSync - Polarity** and **VSyn - Polarity** fields to **neg. edge trig** (negative edge trigger).

6. Select the **Video Timing** tab. Adjust the horizontal back and front porches to match the camera's horizontal frequency (15.734 Hz). To do so, click on the back porch **BPorch - pclk** field and use the up or down arrows to adjust the horizontal frequency. For 15.734 Hz, the back porch value should be 50 pclk.

Here, you are removing or adding pixels in the back porch until the horizontal frequency displayed in the **Frequency** field is at the required value.

Now that the horizontal timings are set, we will interactively test and fine-tune our DCF.

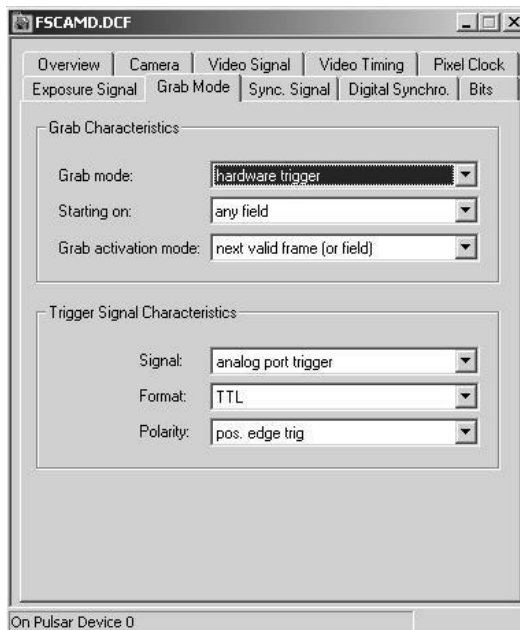
7. Start a continuous grab by clicking on the **Continuous Grab** button.
8. Your image will not be properly centered, and a stripe of black pixels will be visible on both sides of the image. This means that the horizontal and vertical back and front porches have to be adjusted.
 - a. To center your image horizontally, enable the **Lock Active and Total** check box in the **Horizontal** section. Then, click on the **BPorch - pclk** field and use the up and down arrows to fine-tune the image you are grabbing. The final value for the back porch should be 59, and the final value for the front porch should be 15.
 - b. To center your image vertically, enable the **Lock Active and Total** check box in the **Vertical** section. This "locks" the number of lines in the back and front porches to 525 as you are adjusting the blanking lines in the back and front porches.
 - c. Click on the **Back Porch - lines** field and increase its value until your image is aligned vertically. The final value for the back porch should be 33 lines.

9. Update the camera information in the **Camera** tab. In the **Camera name** field, type "*FSCAM, high resolution, non-interlaced, digital*". Also update the **Comments** field, if necessary.
10. Once you are satisfied with the quality of the grabbed image, save your new DCF under the name *FSCAMD.DCF*. To do so, click on the DCF with the right mouse button and select **Save As**, or use **File - Save As** from the menu bar.
11. Stop the continuous grab by clicking on the **Halt Grab** button.

Asynchronous reset trigger

Now, we'll modify the DCF we just created for digital data (*FSCAMD.DCF*) to use the camera's external asynchronous trigger mode and the automatic exposure (shutter) control.

- ❖ Using the camera's external asynchronous trigger mode resets the vertical synchronization and grabs the next frame immediately.
1. Supply a trigger source to the analog trigger input of the Pulsar, e.g.: 1 pulse/second \square . To also send this trigger to the camera, connect the **Exposure1** or **Exposure2** signal of the Pulsar to the camera's trigger (vertical reset or init) input. In this case, we will use the **Exposure2** output signal because **Timer2** (connected to **Exposure 2**) will be used later to control the exposure.
 2. Set the camera switch to ASY (asynchronous reset mode) and the shutter control to Position 7, which corresponds to a self-generated exposure time of 1/250 sec. for our example camera.
 3. Choose the **Grab Mode** tab.




In the **Grab Characteristics** section:

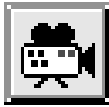
- a. Set the **Grab mode** to **hardware trigger**.
- b. Set **Starting on** to **any field**.
- c. Set the **Grab activation mode** to **asynchronous reset**.

In the **Trigger Signal Characteristics** section, set **Signal** to **Analog port trigger**.

4. As mentioned earlier, the trigger pulse received by the Pulsar must be sent to the camera. To do this, we will use the **Timer 2** output (**Exposure2**) to reset the camera's vertical sync.

Click on the **Exposure Signal** tab.

- a. Click on the **Timer 2** sub-tab.
 - Set **Mode** to **on trigger event**.
 - In the **Trigger Info** section, set **Signal** to **Timer 1 output**. Here, **Timer 1** and **Timer 2** are cascaded.
 - In the **Active Duration** section, set **Pulse Width** to 0.0002 seconds and **Delay** to 0 seconds.
 - In the **Exposure Signal Info** section, set **Polarity** to **neg. edge trigger**. This is to generate a negative pulse. 
- b. Click on the **Timer 1** sub-tab.
 - Set **Mode** to **on trigger event**.
 - In the **Trigger Info** section, set **Signal** to external signal. This pulse on **Timer 1** will start **Timer 2** when it receives an external trigger pulse.
 - In the **Active Duration** section, set **Pulse Width** to 0.0002 seconds.



5. Start a continuous grab by clicking on the **Continuous Grab** button, and adjust the DCF's video timings, if necessary.
6. Update the camera information in the **Camera** tab. In the **Camera name** field, type *"FSCAM, high resolution, digital, non-interlaced, trigger"*. Also update the **Comments** field, if necessary.
7. Save your DCF under the name *FSCAMT.DCF*. To do so, click on the DCF with the right mouse button and select **Save As** or use **File - Save As** from the menu bar.



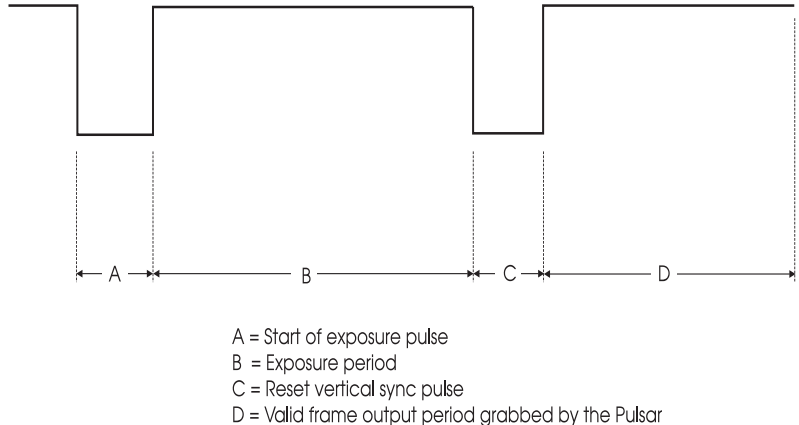
8. Stop the grab by clicking on the **Halt Grab** button.

Exposure settings

Now, we'll modify the DCF we just created for asynchronous reset trigger mode (*FSCAMT.DCF*) to dynamically control the camera's CCD shutter (exposure). To do this, we have to specify an exposure signal in the DCF.

1. Select the double pulse shutter mode of our example camera by setting the camera's shutter selector to Position 9.

In this mode, the first negative pulse received by the camera triggers the exposure, and the second pulse resets the vertical sync and outputs the exposed frame, as shown below.



2. Click on the **Exposure Signal** tab.
 - a. Click on the **Options** sub-tab. Enable the **XOR timer 1 and timer 2 output** option. This combines the signals generated by the two timers and creates the required waveform (with two negative pulses).
 - b. Click on the **Timer 2** sub-tab. In the **Active Duration** section, set **Delay** to 0.010 seconds. This represents the exposure period.
3. Start a continuous grab by clicking on the **Continuous Grab** button.
4. Modify the exposure time by varying the delay value. For example, setting the **Delay** to 0.02 seconds should make the image brighter. Note that this delay is software

programmable and can be changed outside of Intellicam using, for example, the MIL command **MdigControl** (in an external program).

5. Update the camera information in the **Camera** tab. In the **Camera name** field, type *"FSCAM, high resolution, digital, non-interlaced, trigger, exposure"*. Also update the **Comments** field, if necessary.
6. Save your DCF under the name *FSCAMTE.DCF*. To do so, click on the DCF with the right mouse button and select **Save As** or use **File - Save As** from the menu bar.
7. Stop the grab by clicking on the **Halt Grab** button.

Interfacing a line scan camera

In this example, our target line scan camera is the fictitious **LSCAM**. This is a linear camera with a line trigger (2988 x 480, dual 8-bit tap). The video timings are as follows: digital, non-interlaced, monochrome, 8-bit, continuous trigger, with a 15.000 MHz pixel clock. The line scan rate is variable. With this camera, two 8-bit pixels (forming a 16-bit word) are output for each pixel clock. Our target digitizer is the **Matrox Pulsar**.

❖ Such cameras are often called dual-tap cameras.

To interface the line scan camera with the Pulsar digitizer, you must first connect your line scan camera to the RS-422 digitizer module, using a custom, digital input cable (see the *Technical Information* chapter of the hardware manual for your specific board).

An external trigger signal is sent to the Pulsar which, in turn, sends a trigger to the camera, as follows:

1. Supply a trigger source to the analog trigger input of the Pulsar, e.g.: 100 pulses/second \square L. To send the exposure and the trigger to the camera, connect the **Exposure1** signal (and the **Exposure2** signal, depending on the camera) of the Pulsar to the line trigger input of the camera. With our LSCAM camera, we will use **Exposure1** to control the exposure time (CCD charge) and **Exposure 2** to trigger the start of the video for each line.
2. Click on the **System Selection** button or choose **System Selection** from the **Options** menu. Then choose **Pulsar Device 0** as your target system.





3. Create a new DCF initialized with an existing DCF:
 - a. Click on the **New DCF** button or choose **New** from the **File** menu, and then make sure the **Digitizer Configuration Format** tab is active.
 - b. Choose RS-170 or CCIR, or **Browse** to find an existing DCF that is similar to the configuration of the **LSCAM**.
 - c. Click on **OK**.
4. Choose the **Camera** tab. From the **Type** field, choose **Line Scan**.
5. Choose the **Sync. signal** tab. From the **Source** field, choose **Camera**. From the **Format** field, choose **digital**. From the **Synchronization signal available** field, choose **Hsync**.
6. Choose the **Digital Synchro.** tab. In the **Hsync** section, enable the **Input active** check box. From the **Format** field, choose **RS-422**, if it is not already selected. From the **Polarity** field, choose **pos. edge trig**.
7. Choose the **Video Signal** tab. From the **Type** field, choose **digital**. From the **Data bus width** field, choose **16 bits** (dual 8-bit inputs).
8. Choose the **Pixel clock** tab. In the **Pixel clock** field, type in the new frequency value: 15.000 MHz. From the **External clock signal** field, choose **Generated by digitizer and returned by camera**. From the **I/O Characteristics Input** field, choose **1*pixel clock** for the frequency, **RS-422** for the format, and **pos. edge trig.** for the polarity. From the **I/O Characteristics Output** field, choose **2*pixel clock** for the frequency (to return 2 times the input clock to the camera), **RS-422** for the format, and **pos. edge trig.** for the polarity.

9. Choose the **Video timing** tab. Select the camera resolution by adjusting the horizontal timings of the DCF:
 - a. From the **Vertical timing type** field, choose **no vertical**. In the **Horizontal** section, set the **Active - pclk** field to the new horizontal resolution: 2988.
 - b. Adjust the back porch and the front porch to best match the camera's horizontal frequency (4.9 KHz).
 - c. In the **Vertical** section, set the **Active - lines** values field to the new vertical resolution: 480 (to complete the window grab area). This might vary, as needed.
10. Choose the **Grab mode** tab.
 - a. In the **Grab characteristics** section, choose **hardware trigger** from the **Grab mode** field. From the **Starting on** field, choose **any field**. From the **Grab activation mode** field, choose **asynchronous reset**.
 - b. In the **Trigger signal characteristics** section, choose **Ext. hsync of dig. port** from the **Signal** field. From the **Polarity** field, choose **pos. edge trig**.
11. Choose the **Exposure signal** tab and then select the **Timer 1** sub-tab.
 - a. In the **Generation** section, choose **on trigger event** from the **Mode** field.
 - b. In the **Trigger Info** section, choose **extern. sig** from the **Signal** field. From the **Format** field, choose **TTL**. From the **Polarity** field, choose **pos. edge trig**.
 - c. In the **Active duration** section, set **Pulse width** to 4e-03 seconds and set **Delay** to 0.0 seconds.
 - d. In the **Exposure clock** section, choose **Synchr. to pixel clk** from the **Type** field. From the **Division factor** field, choose **Base clock/1**, if it is not already selected.

The **Timer 1** signal is sent to the camera to initiate the CCD charge (beginning of **exposure time**).

12. Choose the **Timer 2** sub-tab.
 - a. In the **Generation** section, choose **on trigger event** from the **Mode** field.
 - b. In the **Exposure Signal Info** section, choose **RS-422** from the **Format** field. From the **Polarity** field, choose **pos. edge trig.**
 - c. In the **Trigger Info** section, choose **timer 1 output** from the **Signal** field. From the **Polarity** field, choose **pos. edge trig.**
 - d. In the **Active duration** section, set **Pulse width** to 3.5e-005 seconds and set **Delay** to 3.7e-003 seconds.
 - e. In the **Exposure clock** section, choose **Synchr. to pixel clk** from the **Type** field. From the **Division Factor** field, choose **Base clock/1**, if it is not already selected.

The **Timer 2** signal is sent to the camera to start the video (beginning of **video valid**).

13. Start a continuous grab by clicking on the **Continuous grab** button.
14. Your image will not be properly centered, and a stripe of black pixels will be visible on both sides of the image. This means that both the **back porch** and the **front porch** have to be adjusted. Adjust the timings until you are satisfied with the results.
15. Save your DCF under the name *LSCAM.DCF*. To do so, click on the DCF with the right mouse button and select **Save As** or use **File - Save As** from the menu bar.
16. Stop the grab by clicking on the **Halt Grab** button.

Chapter 5: The Matrox Imaging Library Interpreter

The Matrox Imaging Library (MIL) Interpreter lets you use the Matrox Imaging Library functions without having to use a C compiler.

What is the Matrox Imaging Library Interpreter?

Intellicam includes an interactive, command-line interpreter (called MILINTER) that gives you access to all the functions provided with MIL (or MIL-Lite). MILINTER enables you to experiment with the MIL functions without having to recompile every time a change is made.

Easy to use

Because the syntax of MILINTER is similar to C language grammar, experimenting with the functions available in the Matrox Imaging Library is easy for programmers experienced with C. This similarity of programming syntax means that you can develop C programs based on command lists.

If you need information about a specific MIL command parameter, an on-line help is provided. Moreover, if you mistype a command name or forget to include a necessary parameter, you will be given a short explanation of the error that has occurred, along with a description of the particular command and the parameters that must be passed to it.

Getting started with MILINTER

MILINTER overview

MILINTER is automatically launched when Intellicam is launched, giving you access to all the library commands in the Intellicam application environment. A dedicated window acts as a command-line interface to MILINTER.

Invoking MILINTER

When MILINTER starts up (which occurs when Intellicam starts up), an interpreter start-up file (*milinter.ini*) is executed to initialize the interpreter state and the macros. This file is, in essence, a command list file that can include any MILINTER command, or call other command lists.

You can change the contents of this interpreter start-up file. You can also change its name and/or location at any time, using

Interpreter - Preferences or **Options - Preferences** (**Interpreter** tab) from the menu. Select the **Startup file** category and type the new location and/or the filename. The new filename and/or location will be used the next time you launch Intellicam.



To start MILINTER, click on the **MIL Interpreter** button or choose **Interpreter - Command Window** from the menu. This opens a command-line window where you can interactively access all the commands provided with the Matrox Imaging Library.

*Directories for help files
and image loading*

You can set up the directory in which you want to keep the MILINTER help files and a default directory in which to store your image files, using **Interpreter - Preferences** or **Options - Preferences (Interpreter tab)** from the menu. Select the **File path information** category and type the location and the filename.

The MILINTER command-line window opens and a list of pre-defined buffer names for the active image is displayed. Images can be accessed using their names. For example, entering *MbufClear Image1 128* will clear the contents of the "Image1" document to the value 128 (gray). The *ACTIM* and *ACTCH* macros always point to the active image document and to its associated active child area, respectively. As in any image document, a child area can be defined by double clicking at the desired position in the image document and dragging the child area to the desired size.

MILINTER tool bar buttons

You can access MILINTER quickly using the interpreter tool bar buttons:

- The interpreter command window button (**MIL Interpreter**) is used to show or hide the MILINTER command window.
- The interpreter user-defined command line buttons (**Interpreter Command 0 to 2**) let you define frequently used commands to launch them quickly, without having to type in the command line.

Such user-defined command lines can be changed at any time by selecting **Interpreter - Preferences - Category: Command Buttons** from the menu. Type in the line you require for any of the three buttons and click on **OK** to accept the changes.

Using MILINTER

As a general rule, you can use any MIL (or MIL-Lite) command (depending on which version of Intellicam you have installed) with MILINTER.

Intellicam and MILINTER run in two separate threads, and can perform operations independently of each other; however, objects created using Intellicam can be accessed through MILINTER. To perform actions on these Intellicam objects, make sure that you have the corresponding MIL IDs. Intellicam provides you with its MIL object IDs via macros that are kept up-to-date in MILINTER. If you use Intellicam-allocated objects, you must make sure that you do not interfere with Intellicam's usage of that object. For example, objects allocated by Intellicam must not be freed by MILINTER.

- ❖ MIL objects that you allocate with MILINTER are not known to Intellicam and can be used in MILINTER only. Moreover, you must free these objects when they are no longer used since Intellicam will not free them automatically.

Example

Each image opened using Intellicam has a corresponding macro to access the MIL image buffer ID created by Intellicam. For example, an image file (*board.mim*) loaded in Intellicam can be accessed in MILINTER by typing in the filename, without the extension. If you define a child area using your mouse, you can use this region by typing in the filename followed by "CH" (*boardCH*).

Following is a step-by-step example:

1. Load Intellicam.
2. Select a system.
3. Open an existing image
(\MIL\EXAMPLES\BOARD.MIM)
4. Open the **Interpreter Command** window (**Interpreter - Command Window**).

A short text explains that *ACTIM* is now pointing to *board.mim* and *ACTCH* to its child area.

5. Define a child area by double clicking on the required child area's origin point and then dragging the mouse to the required end point.
6. Type the following in the MILINTER window to clear the child area: "**MbufClear ACTCH 0**".
7. Create an image document (640 x 480 x 8-bit). The new window, called **Image1**, can be accessed using its name and *ACTIM* when active.
8. Copy the modified board image into this new image by typing **MbufCopy board image1** in the **MILINTER Command** window.

Intellicam macros

Intellicam includes macros for opened image documents, allocated Intellicam system objects, and allocated Intellicam application objects.

Opened image documents

Intellicam document	MILINTER MIL ID macro
<DocName>	For the whole MIL image buffer
<DocName>CH	For the child area of the MIL image buffer
<DocName>DISP	For the MIL display object used to display the MIL image buffer

Examples:

Image1	Image1	Image1CH	Image1DISP
Cell.mim	Cell	CellCH	CellDISP
SunSet.tif	SunSet	SunSetCH	SunSetDISP
SunSet.tif2	SunSet	SunSetCH	SunSetDISP

Currently active image document can be accessed using:

ActIM	For the whole MIL image buffer of the active image buffer
ActCH	For the MIL child buffer of the active image view
ActIMDISP	For the MIL display object used to display the active MIL image buffer

Allocated Intellicam system objects

Intellicam system	MILINTER ID macro
<BoardName>SYS_<N>	For the MIL system object
<BoardName>DIG_<N>	For the MIL digitizer of the system object
<BoardName>DISP_<N>	For the MIL display of the system object (default display)

Examples:

```
Meteor device 0 MeteorSYS_0 MeteorDIG_0 MeteorDISP_0
Pulsar device 2 PulsarSYS_2 PulsarDIG_2 PulsarDISP_2
Host system    HostSYS_0    HostDIG_0    HostDIPS_0
device 0
```

Currently active system object can be accessed using:

```
ActSYS    For the active (selected MIL system object)
ActDIG    For the active MIL digitizer of the system object
ActDISP   For the active MIL display of the system object
```

Allocated Intellicam application object

Intellicam application	MILINTER ID macro
<AppName>IntelcamAPP	For the MIL application object

Examples:

```
Intellicam application IntelcamAPP    Currently active
                                       application object
ActAPP                                For the MIL
                                       application object of
                                       Intellicam
```

Matrox Imaging Library (MIL) examples

A sample program, *mstart.cl*, allows you to test the installation and become familiar with running a MIL application. This test program opens communication with the current system, initializes the global state of MIL, displays a welcoming message, pauses, and then closes communication with the board.

Running MIL command lists

1. Load the sample program from disk by typing the following command at the MILINTER prompt:

```
loadcl mstart "mil\intelcam\inter\examples\mstart.cl"
```

2. Then, run it by typing the following at the MILINTER prompt:

```
mstart
```

3. You can now edit this example by typing the following line at the prompt:

```
editcl mstart
```

For example, you can change the text to be printed.

If you want to save your edited example, use the **saveCL mstart "mstart2.cl"** command.

About the examples

The examples found in the Matrox Imaging Library package have been translated into MILINTER syntax. They can be run using the same procedure as that used for the sample program *mstart.cl*, described above. These examples can help you learn MIL functions without having to set up a programming environment.

To run all the MIL examples one after the other, load and run the *exrun.cl* command list.

Setting up command lists

The *milsetup.cl* file is a command list used to set up the environment before any MIL command list examples are run. The purpose of this command list is to define the default system, camera, image sizes, and attributes that will be used by any MIL command list example when allocating default objects. These defaults are used by two other command lists, *MappAllocDefault* and *MappFreeDefault*, which are described in detail in *Matrox Imaging Library standard types and macros* later in this chapter.

Editing *milsetup.cl*

You can change the system, camera, or display used by the sample programs by editing the *milsetup.cl* text file. Note that the *milsetup.cl* configuration file is located in the *MIL\INTELCAM\INTER* directory.

Notational conventions

"char"	Characters between quotation marks are used to define a fixed value item. Example: "6"
<item>	An item enclosed in <> indicates that you must supply a parameter. Example: loadmacro <filename>
[item]	An item enclosed in square brackets indicates an optional parameter. Example: execl try.cl [parameter]
item...	Three dots following an item indicate that more items with the same form may be entered. Example: execcl <filename> [parameter...]
key names	Key names separated by a plus sign (+) indicate that you must press both keys at the same time. Example: ctrl+left, etc.

Matrox Imaging Library (MIL) standard types and macros

MILINTER provides some standard Matrox Imaging Library (MIL) types and macros.

MIL standard variable types

The standard MIL variable types are: MIL_ID, CHAR, SHORT, LONG, and DOUBLE.

MIL_ID

■ Synopsis:

The general purpose variable type used to declare MIL objects before allocation.

■ Usage:

MIL_ID	Variable declaration
<identifier>	
<identifier>	To pass the value to a function
&<identifier>	To pass the address to a function

■ Example:

```
; This example allocates an image buffer and draws
; a line in the latter, using the default graphics context

MIL_ID MillImage
MbufAlloc2d ACTSYS 640 480 8 (M_IMAGE+M_DISP) &MillImage
MdispSelect ActDISP MillImage
MbufClear MillImage
MgraLine M_DEFAULT MillImage 0 0 100 100
```

CHAR, SHORT, LONG, DOUBLE■ **Synopsis:**

The CHAR, SHORT, LONG, and DOUBLE types are analogous to the C language variable types.

■ **Usage:**

CHAR <identifier>	Variable declaration
SHORT <identifier>	Variable declaration
LONG <identifier>	Variable declaration
DOUBLE <identifier>	Variable declaration
<identifier>	To pass the value to a function
&<identifier>	To pass the address to a function
SET <identifier>	To assign a value to the variable
<value>	
GET <identifier>	To print the variable content

■ **Example:**

```
;Select the foreground color to be used
; with the default ;graphics context
LONG MilColor
SETMilColor 128
MgraColor M_DEFAULT MilColor

;Draw an elliptical arc in the MillImage buffer.
DOUBLE Mil Angle
SETMilAngle
SETMilAngle 45.5
MgraArc M_DEFAULT MillImage 100 100 50 50 0.0 MilAngle
;Print out the width of the MillImage buffer
LONG MilWidth
MbufInquire MillImage M_SIZE_X &MilWidth
GETMilWidth

;Free the previously allocated image buffer
MbufFree MillImage
```

MIL macros

The standard MILINTER macros include *MappAllocDefault*, and *MappFreeDefault*.

MappAllocDefault

MappAllocDefault allocates an application, a system, digitizer, display and buffer using default definitions from the *milsetup.cl* command list. Since MILINTER is embedded in Intellicam, these default objects are, in general, not actually allocated; instead, the ID of the current Intellicam MIL objects are returned. This allows object sharing between Intellicam and MILINTER, that is, the current system and related object IDs in Intellicam, to be used in MILINTER. When you use these objects in MILINTER, you will simply be manipulating Intellicam objects. Whenever these objects are not already allocated by Intellicam, this macro will actually allocate them, using the *milsetup.cl* information.

■ Synopsis:

This macro sets up the requested MIL and processing environments using the defaults specified in the *milsetup.cl* file. It can allocate and initialize a MIL application, allocate the system that will receive the MIL commands, allocate the digitizer and display, and allocate and clear a displayable image buffer on this target system, depending on what is requested.

■ Usage:

```
MappAllocDefault  LONG InitFlag
                  MIL_ID *ApplicationPtr
                  MIL_ID *SystemIdPtr
                  MIL_ID *DisplayIdPtr
                  MIL_ID *DigitizerIdPtr
                  MIL_ID *ImageBufIdPtr
```

■ Examples:

```
;Allocate an application, open communication
;with the system, allocate the display, allocate
;the default image buffer and display it.
```

```
MIL_ID MilApplication
MIL_ID MilSystem
MIL_ID MilDisplay
MIL_ID MillImage
```

```
MappAllocDefault M_DEFAULT &MilApplication &MilSystem
&MilDisplay M_NULL &MillImage
```

```
;Performing the above setup without allocating the display
```

```
MIL_ID MilApplication
MIL_ID MilSystem
MIL_ID MillImage
```

```
MappAllocDefault M_DEFAULT &MilApplication &MilSystem M_NULL
M-NULL &MillImage
```

MappFreeDefault

MappFreeDefault is responsible for freeing objects allocated with *MappAllocDefault*. Since *MappAllocDefault* often returns object IDs that are in fact Intellicam objects, the "freeing" operation is not actually done; it is simply ignored. If an object is not shared with Intellicam, *MappAllocDefault* will actually free the object.

■ Synopsis:

This macro frees the defaults which were allocated using the *MappAllocDefault* command.

■ Usage:

```
MappFreeDefault      MIL_ID ApplicationId
                     MIL_ID SystemId
                     MIL_ID DisplayId
                     MIL_ID DigitizerId
                     MIL_ID ImageBufId
```

■ Examples:

```
;Free the defaults allocated in the MappAllocDefault's first  
example
```

```
MappFreeDefault MilApplication MilSystem M-NULL MilDisplay  
Millmage
```

Help

For a brief description of a command, including its parameters and their data types, enter the following at the command line:

```
help [identifier]
```

If a command identifier is entered, the grammatical structure, usage, and parameter data type for the command will be printed. By entering the first few letters of a series of commands, MILINTER will print out information about all the commands that start with those letters. For example, by typing "Help Mbuf", MILINTER will list all functions that start with "Mbuf". This will give you a list of most of the functions related to MIL data buffer control.

Command definition

Once initialization is complete, you are asked to type in a command at the > prompt.

MILINTER provides two types of commands: internal commands (commands that are already defined in the interpreter and do not directly call MIL) and MIL library functions. At the prompt, you will have to type something similar to the line below:

`<identifier>[parameter...]`

<Identifier> refers to either a command internal to the interpreter, a function of the library, a macro, or the name of a command list (see the *Interpreter Syntax* section later in this chapter for the identifier syntax). The list of all reserved command identifiers can be obtained by typing the command *RESERVED* at the prompt.

[Parameter...] refers to a list of parameters corresponding to the command entered.

The command identifier and all the parameters on the command line must be separated by at least one space (spaces between quotation marks are ignored).

Other considerations

You can recall a previous command by typing in the first few letters of the required command and then pressing **F8**.

MILINTER is **not** case-sensitive, which means you can enter commands either in upper case or lower case characters, or a mixture of both.

MILINTER errors

If the command could not be executed because an error occurred, an error message will be displayed.

Two types of error can occur when using MILINTER:

- A MILINTER error caused when a parameter is omitted, an unknown command used, or typo made. MILINTER prints this type of error in the command window.
- A MIL error caused when a call to MIL is made using a MILINTER command line. When this happens, a message box opens to alert you of the error: "Intellicam - MILINTER error! (code=0xnn)". More information regarding the error follows this message. Click on **OK** to acknowledge the error, and continue using MILINTER.

If the command was processed correctly, the return value will be printed (if there is a return value), and you will be prompted for another command.

Parameter definition

MILINTER has a very flexible way of defining parameters. A parameter can be defined as a constant, a numerical expression, a string, or a variable identifier.

The parameter syntax is as follows:

```
parameter = <constant or numerical expression or string or
identifier>
```

Constant

A constant can be either an integer, a floating point, or a character:

```
const = <int_const or float_const or char_const>
```

By default, all integer constants are decimal. It is also possible to enter hexadecimal, binary and double values on the command line. Hexadecimal values have a prefix of 0x (or 0X), binary values have a prefix of 0b (or 0B), and DOUBLE values have a prefix of 0d (or 0D).

```
const =      <<dec_digits> or
              <0B or 0b> <bin_digits> or
              <0X or 0x> <hex_digits> or
              <0D or 0d> <double_digits>
```

For example: 10, 0b1010, 0xA, 10.0, and 0d10 are 5 different ways of representing the decimal value 10.

Character constants can also be used by putting any single representable character between single quotation marks. A representable character is any character that is visible on the screen, such as any letter of the alphabet or any character.

```
char_const = <' char '>
```


Numerical expression

A numerical expression is an arithmetic equation that contains values and operators that result in a number.

For example: -4, (4+6), (4+(0x10>>2)), (var+6)

The values can be constant or variable, and operators can be monadic (one operand) or dyadic (two operands).

Monadic operator

A monadic operator has a single operand and results in a number.

For example, " - " in -4 results in the negative of 4.

■ **Syntax:** monadic_op<value>

The available operators are: monadic_op = <! or ~ or - or +>

Operator	Function
!	Logical NOT
~	Bitwise complement
-	Unary MINUS
+	Unary PLUS

Dyadic operator

A dyadic operator has two operands and results in a number.

For example, (4+4) results in 8.

■ **Syntax:** <value> dyadic_op <value>

The available dyadic operators are:

Dyadic_op = <* or / or % or + or - or << or >> or == or < or > or
! or <= or >= or & or | or ^ or && or || >

All these operators conform to C programming grammar.

Operator	Function
*	Multiplication
/	Division
%	Remainder
+	Addition
-	Subtraction
<<	Left shift
>>	Right shift
==	Equality
<	Less than
>	Greater than
!=	Inequality
<=	Less than or equal to
>=	Greater than or equal to
&	Bitwise AND
	Bitwise OR
^	Bitwise XOR
&&	Logical AND
	Logical OR
&&	Logical AND
	Logical OR

Here are some examples of numerical expressions:

```
-123
0x1FFB0
( (0x03F0 >> 2) * 3 )
-( (3 + (0x5F & 'A') * 5) && ! ('*' >= 0b1111011) )
```

All expressions are evaluated from right to left, with no precedence on the operations, except for those enclosed in parentheses. All the values are converted to 32-bit signed integers, and all the operations are performed on 32-bit signed integers, except if a floating point value is used in the expression. MILINTER does **not** detect overflows on operations and, as such, will allow erroneous results to be passed to functions. A *"Parameter out of range"* error will occur if the input value for a given operation does not fall within the limits permitted for this kind of parameter.

Operations are also possible on received external parameters that are recovered using the #1 ... #9 expressions (see the *Identifier* section for more information on this type of parameter), and on '_' expressions (see the *Loop* section for more information on this type of parameter).

Command names

Command names used as a number are defined as follows:

```
command = <identifier> [parameter...]
```

Example

Here is an example of the usage of the *SETNBR* command. This function assigns a numerical value to a specified identifier.

```
> SetNbrn3
> SAetNbr m2
> SetNbr n ([GetNbr n]+[GetNbr m])
> GetNbr n
Return DWORD=5
```

In this example, the command *GETNBR* returns a numeric value that can be used as a constant in a numerical expression or used directly as a parameter. All the commands used in a parameter definition must return a value. This value can be a number or a string, but a string cannot be used in a numerical expression.

String

As mentioned, a parameter can also be a string defined as follows:

```
string = <" [char...] " or "[ command ]">
```

You can use quotation marks (") to pass a whole string as a parameter. When a string is enclosed in quotation marks, it is considered as one parameter. All characters enclosed in quotation marks are not converted to lower case, and macros are **not** expanded, but #1 ... #9 expressions are replaced by their corresponding parameter values.

Note that a string can also be a command, but the command must return a string.

Example

Here is an example of the string parameter used with the *SETSTR* command. This function assigns a string value to a specified identifier.

```
> SetStr s0"ABCD"
> SetStr s1"EFGH"
> GetStr s0
Return STRING=ABCD
>SetStr s0 [GetStr s1]
>GetStr s0
Return STRING=EFGH
```

The interpreter checks for parameter type mismatches. If you try to pass a string parameter where only a numerical parameter can be specified, MILINTER will print an error message.

Identifier

An identifier is a name given to a variable. The identifier syntax is a letter followed by some other letters or numbers (see the *Interpreter Syntax* section). As demonstrated in the previous example, you can also use an identifier as a parameter.

Editing

MILINTER provides a line editor with which to enter commands. The keys that provide special functionality are:

Left arrow	Moves the cursor left one character; stops at the beginning of the line.
Right arrow	Moves the cursor right one character; stops at the end of the line.
Home	Moves the cursor to the beginning of the line.
End	Moves the cursor to the end of the line.
Ctrl+Left arrow	Moves the cursor left to the start of the nearest word.
Ctrl+Right arrow	Moves the cursor right to the start of the nearest word.
Ctrl+PgUp	Clears the <i>History</i> buffer. See the <i>History</i> section.
Up arrow	Moves up in the <i>History</i> buffer. See the <i>History</i> section.
Down arrow	Moves down in the <i>History</i> buffer. See the <i>History</i> section.
Esc	Erases the entire current line and moves the cursor to the beginning of the line.
Delete	Erases the character at the current cursor location; moves the rest of the line one character to the left.
Backspace	Erases the character to the left of the cursor; moves the cursor and the rest of the line one character to the left.
Ctrl+End	Deletes from the cursor to the end of the line.
Ins(ert)	Toggles between insert and overwrite mode and changes the cursor shape accordingly (see below).
Enter	Ends the user input and calls the MILINTER parser to process the command line (the cursor can be at any position on the line).

*Insert and
overwrite modes*

In insert mode, each character entered moves the rest of the line (to the right of the cursor) one character to the right to make room for the new character. In overwrite mode, any character entered replaces the character currently at the cursor position.

History

All the commands you enter are saved in a list, called a History buffer, that contains a maximum of 50 entries. Every time a new command is entered, it is placed at the end of the list. When the list is full, the next entry overwrites the entry at the bottom of the list.

The "Up" arrow key recalls the previous command from the list. The "Down" arrow key recalls the next command from the list. The "Ctrl+PgUp" key clears all the commands from the list. The cursor can be at any position on the line when these keys are pressed.

Once a command has been recalled, it can be executed by pressing "Enter", or it can be edited just like any normal command. If the command is not edited before being executed, it will not be placed in the list again. If changes are made, it will be added to the end of the list.

If the "Enter" key is pressed when the command line is empty, the pointer is repositioned at the end of the list.

Manipulation of variables

Manipulation of numbers

MILINTER provides some simple commands for integer number manipulation. All the numbers are converted to 32-bit signed integers, and all the operations are performed on 32-bit signed integers. A declaration of an unsigned type is not permitted. Therefore, SHORT variables cannot take on values greater than 0x7fff. Overflows on operations are **not** detected. You must therefore assure that values passed to commands are within the parameters' ranges. A *"Parameter out of range"* error will occur if the result of the operation does not fall within the limits permitted for this type of parameter.

The commands for number manipulation are:

- **ABS <value>**
Returns the absolute value of a unary expression.
- **CLEARNBR <identifier>**
Erases the specified identifier from the number list and clears the memory associated with it. The identifier must already have been defined by the *SETNBR* command in the number list. If no identifier is specified, *CLEARNBR* will erase all the identifiers in the number list and clear the memory associated with them.
- **DBLTOSTR <number> <nb_of_digits>**
Converts a floating-point number (double) into a string according to the number of significant digits required. Returns a pointer to the string of digits.
- **GETNBR <identifier>**
Returns the value of the number identifier. If the number identifier is not in the number list (i.e., no value was previously assigned to this identifier by the *SETNBR* command), the value 0 is returned.
- **NBRTOSTR <value> <base>**
Converts a numerical expression value in a string according to the specified base parameter. The base parameter can be a numerical expression with a value between 2 and 35.

- **RAND**

Returns a random value between 0 and 32767.

- **SETNBR <identifier> <value>**

Assigns a numerical expression value to the specified identifier and adds this identifier to the number list.

Example:

```
> SetNbr n3
> SetNbr 2
> Set n([GetNbr n]+[GetNbr m])
> GetNbr n
Return DWORD=5
```

Manipulation of strings

The interpreter also supports commands for string manipulation. You can use quotation marks (") to pass a whole string as a parameter. When a string is enclosed in quotation marks, it is considered as one parameter (even if there are spaces). Such strings are **not** converted to lower case. Macro identifiers in a string are **not** expanded, but #1 ... #9 expressions are replaced by their corresponding values (see the *Command lists* section for more information on this type of parameter).

A string parameter can also be a command, but this command has to return a string.

The commands for string manipulation are:

- **ADDSTR <string1> <string2>**

Appends *string2* to *string1* and returns the resulting string (maximum resulting string length = 1024 characters).

- **CLEARSTR <identifier>**

Erases the specified identifier from the string list and clears the memory associated with it. The identifier must already have been defined by the *SETSTR* command in the string list. If no identifier is specified, *CLEARSTR* will erase all the identifiers in the string list and clear the memory associated with them.

- **CMPSTR <string1> <string2>**
Compares *string1* and *string2* and returns -1 if *string1* is less than *string2*, 0 if *string1* is equal to *string2*, and 1 if *string1* is greater than *string2*.
- **GETSTR <identifier>**
Returns the string value of the string identifier. If the string identifier is not in the string list (i.e., no value was previously assigned to this identifier by the *SETSTR* command), a NULL string is returned.
- **LEFTSTR <string> <n>**
Returns the leftmost *n* characters of the string. The parameter *n* is a numerical expression that must be in the 0 to 1024 range.
- **LENSTR <string>**
Returns the number of characters in the string.
- **MIDSTR <string> <n> <m>**
Returns *m* characters from the string beginning at the *n*th character. If *n* is greater than the string length, then a NULL string is returned. The parameter *m* is a numerical expression whose value must be in the 0 to 1024 range. The parameter *n* is a numerical expression whose value must be in the 1 to 1024 range.
- **RIGHTSTR <string> <n>**
Returns the rightmost *n* characters of the string. The parameter *n* is a numerical expression that must be in the 0 to 1024 range.
- **SETSTR<identifier> <string>**
Assigns a string value to the specified identifier and adds this identifier to the string list.
- **STRTONBR <string>**
Converts a string value into a number. The string must be in the following form:
 <" <dec_digits> "> or
 <" <0X or 0x> <hex_digits> ">
 If no conversion can be performed, this command returns 0; otherwise, a 32-bit signed integer is returned.

Here is an example of what you can do by specifying this type of parameter.

Example:

```
> SetStr s0"ABCD"
> SetStr s1"EFGH"
> SetStr s01[AddStr[GetStr s0][GetStr s1]]
> GetStr s01
Return STRING=ABCDEFGH
```

Manipulation of arrays

MILINTER can also manipulate arrays. Four types of array commands are provided:

- Character array commands that allow you to access a linear array of unsigned bytes (0 to 255 range) at any specified index.
- Short array commands that allow you to access a linear array of signed words (16-bit signed integers) at any specified index.
- Long number array commands that allow you to access a linear array of signed double words (32-bit signed integers) at any specified index.
- Double array commands that allow you to access a linear array of double (64 bits) floating-point numbers at any specified index.

The commands for array manipulation are:

- **ALLOCARRAY <identifier> <size>**
Allocates memory for a specified array identifier and adds this identifier to the array list. The *size* parameter is the number of bytes to allocate. It can be any numerical expression. If not enough memory is available, an error message is printed.

■ **CLEARARRAY <identifier>**

Frees the memory associated with a specified identifier and erases the identifier from the array list. The identifier must already have been defined by the *SETARRAY* or *ALLOCARRAY* command in the array list. If no identifier is specified, *CLEARARRAY* erases all the identifiers in the array list and frees their associated memory.

■ **GETARRAY <identifier>**

Returns the address of the array identifier. If the array identifier is not in the array list (i.e. no address was previously assigned to this identifier by the *SETARRAY* or *ALLOCARRAY* command), a NULL address is returned.

■ **GETCHRARRAY <identifier> <index>**

Returns the value of a specified character array identifier at the selected index. The *index* parameter can be any numerical expression. You must ensure that this value does not exceed the size of the specified array.

■ **GETDBARRAY <identifier> <index>**

Returns the value of a specified double number array that has been defined by the *SETARRAY* or *ALLOCARRAY* command in the array list. The *index* parameter can be any numerical expression. You must ensure that this value does not exceed the size of the specified array.

■ **GETNBRARRAY <identifier> <index>**

Returns the value of a specified long number array identifier at the selected index. The *index* parameter can be any numerical expression. You must ensure that this value does not exceed the size of the specified array.

■ **GETSHORTARRAY <identifier> <index>**

Returns the value of a specified short number array identifier at the selected index. The *index* parameter can be any numerical expression. You must ensure that this value does not exceed the size of the specified array.

■ **SETARRAY <identifier> <address>**

Assigns an address to the specified identifier and adds this identifier to the array list. This command can give direct access to any memory space and must therefore be used with some caution.

- **SETPHRARRAY <identifier> <index> <value>**
Assigns a numerical expression value (0 to 255 range) to the specified character array identifier at the selected index. The *index* parameter can be any numerical expression equal to a value within the 0 to 65535 range. You must ensure that this value does not exceed the size of the specified array.
- **SETDBARRAY <identifier> <index> <value>**
Assigns a numerical expression value (64-bit floating point) to the specified number array identifier at the selected index. The *index* parameter can be any numerical expression. You must ensure that this value does not exceed the size of the specified array.
- **SETNBRARRAY <identifier> <index> <value>**
Assigns a numerical expression value (32-bit signed integer) to the specified number array identifier at the selected index. The *index* parameter can be any numerical expression. You must ensure that this value does not exceed the size of the specified array.
- **SETSHORTARRAY <identifier> <index> <value>**
Assigns a numerical expression value (16-bit signed integer) to the specified number array identifier at the selected index. The *index* parameter can be any numerical expression. You must ensure that this value does not exceed the size of the specified array.

Example:

```

> AllocArray table 4
> SetChrArray table 0 30
> SetChrArray table 1 2
> SetChrArray table 2 90
> SetChrArray table 3 16
> GetChrArray table 1
Return UBYTE = 2
> ClearArray table ; Don't forget to free

```

MILINTER macro control commands

A macro is defined as a word that can replace a typed string. There is no limitation on the content of the string, aside from the fact that all the characters must be printable. Macros can be placed anywhere on the command line, and can replace a command, or a series of parameters, or both combined in the same macro. The MILINTER macro control commands follow:

A macro name can be used in another macro definition.

The macro name cannot be a command name, a subset of a command name or a command list name. The two reserved words *MACRO* and *DELMACRO* cannot be used in a macro name or in a macro definition.

Refer to the *Interpreter Syntax* section for information on identifier, filename, and definition syntax.

■ **MACRO [identifier [definition]]**

Assigns a definition to the specified identifier, or redefines the macro identifier if it already exists. If the definition part is omitted, the macro identifier, its definition, and its current expansion are printed. If both the identifier and the definition are omitted, the complete list of currently defined macros, their respective definitions, and expansions are printed.

■ **DELMACRO [identifier]**

Deletes the macro identifier from the macro list or, if no identifier is specified, deletes the entire macro list.

■ **SAVEMACRO <filename>**

Saves the entire macro list (names and definitions), as currently defined in the specified file. Unless only one macro is currently defined, macros cannot be saved individually.

■ **LOADMACRO <filename> <[parameter...]**

Retrieves a macro list (identifiers and definitions) from the specified file. If one or more of the macros retrieved were already defined, the former definitions are deleted and those retrieved become the new definition. See the *File handling* section for more information on files.

Command lists

A command list (often referred to as a script) is a list of commands that can be executed sequentially. MILINTER generally stores command lists in memory.

MILINTER also has commands for storing and retrieving command lists that are stored on disk. You can even execute command lists directly from disk. Whether your command lists are saved in memory or on disk, you can specify parameters when invoking them.

When you write a command list, it is usually useful to include comments. You can do so anywhere by inserting ";" followed by your comment. Comments can contain any character, since anything between the ";" and the next line will be ignored by the interpreter. One command list can call another command list, or can call itself recursively. Parameters not defined when invoking the command list are treated as empty strings.

If the identifier specified when invoking the command list does not exist, the editor creates a new command list using the identifier filename.

You can interrupt the execution of a command list at any time by pressing Ctrl+Break. The MILINTER command list control commands are:

- **DELCL [identifier]**
Removes the identified command list from memory or, if no identifier is specified, all the command lists.
- **EDITCL <identifier>**
Edits a command list in memory. If the command list identifier does not already exist, an empty command list is created, and can be edited. See the *Editing command lists* section for more information.

- **EXECCL <filename> [parameters...]**

Executes a command list in the specified file directly from disk, without loading it into memory. The parameters will be accessible from the command list using the symbols #1, #2, ... and #9, the number corresponding to the position of the desired parameter on the line (similar to batch-file handling of parameters). See the *File handling* section for more information on files.

- **IDENTIFIER [parameter...]**

Executes a command list already in memory. Type in the full identifier of the desired command list. The parameters will be accessible from the command list using the symbols #1, #2, ... and #9, the number corresponding to the position of the desired parameter on the line (similar to batch-file handling of parameters). See the *File handling* section for more information on files.

- **LISTCL [identifier]**

Lists, on-screen, the contents of the identified command list defined in memory. If the identifier is omitted, all the identifiers of the command lists currently defined in memory are listed.

- **LOADCL <identifier> <filename>**

Retrieves a command list from the specified file on disk, loads it in memory, and assigns it the specified identifier.

- **ONBREAK [identifier]**

Sets up a command list identifier that will execute when the Ctrl+Break key is pressed. If the identifier is omitted, the prompt is directly returned without executing a command list.

- **RETURN**

Terminates the execution of a command list in which it appears and returns control to the calling command list or to the interpreter when the command list is directly called from it.

- **SAVECL <identifier> <filename>**

Saves the identified command list in the specified file on disk. A warning mechanism protects the user from overwriting a file that already exists.

Editing command lists

Command lists can be edited and saved using any text editor; however, calling *EDITCL* will access Windows' Notepad editor. Refer to the Windows documentation for a description of the Notepad editor. Also note that command lists edited using *EDITCL* are located in memory, and that **File - Save** only saves them to memory. To save to disk, use **File - Save As** or the **SaveCL** command at the MILINTER Command prompt.

Advanced features

Conditional operation

A conditional operator makes decisions with regards to program flow based on the result of a numerical expression.

Branch programming introduces a new parameter selection. In addition to using a numerical expression, string, or identifier, you can use a command block:

parameter = numerical expression *or* string *or* identifier *or* block>

Where block = <command *or* "{ command... }">

You must have only one command per line (a carriage return is a command separator). The interpreter allows the following conditional operators.

- **IF** <condition> <block>

Executes the following command block, if the condition is true (not 0). The condition can be any numerical expression, where the comparison is not made to a DOUBLE constant or double variable.

- **ELSE** <block>

Executes the following command block, if the condition of the last *IF* command (at the same block level) is false (0).

Example:

```
> SetNbr n 3
> if ( [ getnbr n ] == 4 )
> {
>   echo "n == 4"
> }
> else
> {
>   if ( [ getnbr n ] < 4 )
>     echo "n < 4"
>   else
>     echo "n > 4"
> }
n < 4 >
```

Loop

The user can use loops to execute a certain command a specified number of times or until a particular event is reached.

The user can interrupt the execution of a loop at any time by pressing Ctrl+Break.

The interpreter permits three kinds of loops:

■ **DO <block> <condition>**

The command block of the *DO* loop is executed one or more times until the condition becomes false (0). The condition can be any numerical expression. Execution proceeds as follows:

- The command block is executed.
- The condition is evaluated. If the condition is false, the *DO* loop terminates else the process is repeated, beginning with the first step.

■ **WHILE <condition> <block>**

The command block of the *WHILE* loop is executed zero or more times until the condition becomes false (0). The condition can be any numerical expression, where the comparison is not made to a double constant or double variable. Execution proceeds as follows:

- The condition is evaluated.
- If the condition is false, the *WHILE* loop terminates.
- The command block is executed and the process is repeated, beginning with the first step.

■ **FOR [identifier] <start> <end> <step> <block>**

The command block of the *FOR* loop is executed zero or more times until the condition becomes false (0). The *start*, *end*, and *step* parameter can be any numerical expression equal to a value within the -32768 to 32767 range. Execution proceeds as follows:

- The *start*, *end*, and *step* parameters are evaluated. An internal counter is set to the *start* value. The *end* and the *step* parameters are kept in memory and they are not evaluated during the next process.
- If the *step* value is positive:
 - a. If the counter is greater than or equal to the *end* value, the *FOR* loop terminates.
 - b. The command block is executed.
 - c. The counter is incremented by the *step* value and the process is repeated, beginning at a.
- If the *step* value is negative:
 - a. The counter is decremented by the *step* value.
 - b. If the counter is less or equal to the *end* value, the *FOR* loop terminates.
 - c. The command block is executed and the process is repeated, beginning at step 1.
- If the *step* value is equal to 0, the command block is executed indefinitely.

A counter identifier can also be used as a variable for counter value access in the following command block. If the identifier is omitted, the counter identifier *i* is assumed by default.

To access the counter value in a numerical expression, type "_" followed by the counter identifier.

Example:

```

> for 0 6 2 ; Use the default counter i
> {
>   echoIn "i = "
>   echo [nbrtostr _i 10]
> }
> i = 0
> i = 2
> i = 4
> for x 0 2 1 ; Use counter identifier x
> {
>   echoIn "x = "
>   echo [nbrtostr _x 10]
>   for y 2 0 -1 ; Use counter identifier y
>   {
>     echoIn "y = "
>     echo [nbrtostr _y 10]
>   }
> }
> x = 0
> y = 1
> y = 0
> x = 1
> y = 1
> y = 0

```

Timer

Affecting delays in programming or finding out the execution time of some commands might sometimes prove useful. Here is a list of commands to do so:

- **DELAY <milliseconds>**

Waits a specified number of milliseconds before executing the next command. Note that this command should not be used in conjunction with the *GETTIMER* and *SETTIMER* command.

- **GETTIMER**

Prints, on screen, the time elapsed (in milliseconds) since *SETTIMER* was last called.

- **SETTIMER**

Resets the interpreter's internal timer.

I/O - keyboard and Host display

The interpreter provides the following simple I/O commands to make more interactive command lists:

- **CLEARHOST**
Clears the MILINTER window.
- **ECHO [string]**
Prints the specified string and sets the printing position at the beginning of the next line. If no string is specified, nothing is printed and a line is skipped.
- **ECHOLN [string]**
Prints the specified string and leaves the printing position on the same line. If no string is specified, nothing is printed and the printing position stays the same.
- **SETECHOPOS <x> <y>**
Changes the current printing position to column x (1 to 80) and line y (1 to 25).
- **GETECHOPOS**
Returns the x and y coordinates of the current printing position.
- **INSTR <identifier>**
Receives a string from the keyboard until the "Enter" key is pressed and assigns this string value to the specified identifier.
- **INKEY**
Waits for a keystroke and returns a key code.
- **KEYHIT**
Returns a nonzero value if a key has been pressed. Otherwise, it returns 0.
- **PAUSE [string]**
Prints the specified string (or a standard message if no string is specified) on screen and waits for a keystroke before executing the next command.

File handling

All files saved and retrieved by the interpreter are handled in the same way. They are always saved as commands, in plain ASCII text (in the same way a command is normally typed in using the keyboard). When you use the *SAVEMACRO* command, all the macros are saved in a file. When you use the *SAVECL* command, an entire command list already in memory is saved.

There are three commands you can use to retrieve a file: *EXECCL*, *LOADMACRO* and *LOADCL*.

- ***EXECCL***: This command only executes commands residing on disk, not commands entered using the keyboard.
- ***LOADMACRO***: This command is similar to *LOADCL*, and is used in conjunction with *SAVEMACRO*.
- ***LOADCL***: This command retrieves the list of commands and places it in memory without executing it. By assigning a symbolic name for the command list, you can execute the command list at any time by entering its symbolic name.

When you invoke the interpreter, the first thing it does after initialization is call *EXECCL*, either with the filename passed on the command line as a parameter or with the *milinter.ini* filename. If *milinter.ini* is not found in the current directory, *EXECCL* is simply not executed.

Since all the files are saved in ASCII format, they can be created and/or edited using any text editor that accepts and generates ASCII files.

Miscellaneous commands

This section contains commands that do not fit into previously described categories.

- **CLEARALL**

Re-initializes the three types of lists (array list, string list, and number list).

- **GETSTRPTR <name>**

Returns a pointer to the specified string. This string is usually hard coded into a variable, such as a #define. Either an explicit string or a pointer-to-a-string must be passed to functions that require string arguments. Therefore, this function is useful when a filename is hard coded into a variable used repeatedly in a command list.

- **HELPPATH <path>**

Loads a help path into the interpreter. This path is the location of all the help files you might need. If used in *milinter.ini* or any command list, it overrides the INTER_HLP environment variable.

- **QUIT**

Closes the Command Window and returns to Intellicam.

- **RESERVED**

Lists all the reserved words (internal commands and library function names) and currently defined names (macros and command lists).

- **SILENCE <1 or 0>**

Turns the silence mode on (1) or off (0). When the silence mode is on, the return values of a command are not automatically printed on screen. When the silence mode is on, the interpreter prompt is changed to S>.

- **STRTODBL <string>**

Converts the specified string into a floating-point number (double) and returns its value.

- **TRACE <1 or 0>**

Turns the trace mode on (1) or off (0). The trace mode affects only command lists and loop executions. When trace mode is on, the command line is executed and printed on screen. You must press a key before executing the next command. When trace mode is on, the interpreter prompt is changed to T>.

- **VERBOSE <1 or 0>**

Turns the verbose mode on (1) or off (0). The verbose mode affects only command lists and loop executions. When verbose mode is on, the commands are printed on screen after being executed and the interpreter prompt is changed to V>.

When both Verbose and Trace modes are on, the evaluated command line is also printed after being executed. This line contains the final form of the parameter value and is very useful for command list debugging.

Os shell

The *SYSTEM* command lets you open a command prompt and optionally run a command from the interpreter, without having to leave the interpreter, by entering the following line at the prompt:

system [identifier [arguments...]]

A command identifier can be any command that could normally be entered at the command prompt, i.e., regular commands (TIME, DIR, etc.) and valid program names (with *.exe*, *.com*, or *.bat* extensions) including their arguments.

If no command identifier is specified, the interpreter temporarily gives control to the command prompt, and you can do virtually anything you can normally do at the command prompt level. You can return to the interpreter by typing "EXIT" at the command prompt.

If a command identifier is specified, it is executed. As soon as the program is completed, the interpreter regains control.

Whether a command identifier is specified or not, you will be able to use the SET, PATH and PROMPT commands, but the changes will not be saved when you return to the interpreter.

Error messages

☛ ***"Ambiguous command"***

There were not enough characters supplied to differentiate this command from another.

☛ ***"Cannot modify a command list that is currently being executed"***

Although you can modify a command list when running another command list, you cannot modify a command list that is executing. The command is simply ignored.

☛ ***"Cannot open the file properly"***

The interpreter was not able to open the file properly, either because the file is corrupted or because too many files are already open.

☛ ***"Cannot save the file properly"***

An error was detected while the file was being saved. In other words, the save operation might not have been done properly. The file might be corrupted or unusable.

☛ ***"CMD.EXE cannot be found"***

In Windows NT, COMMAND.COM is called *CMD.EXE*. The *CMD.EXE* cannot be found. The COMSPEC variable should be set to reflect the path where the file can be found.

☛ ***"COMMAND.COM is not executable"***

The *CMD.EXE* file has an invalid format and is not executable.

☛ ***"COMMAND.COM cannot be found"***

The *CMD.EXE* cannot be found. The COMSPEC environment variable should be set to reflect the path where the file can be found.

☛ ***"Command list name already exists"***

The name passed as a parameter is already defined as a command list.

☞ ***"Command list name does not exist"***

The command list name passed as a parameter does not exist.

☞ ***"Command list too big for the editor"***

The command list exceeded the supported size, and thus cannot be edited inside of the interpreter.

☞ ***"Command name already exists"***

The name passed as a parameter already exists as a command name (a command internal to the interpreter, a library command or a user-defined command) or as the first distinctive characters of a command name.

☞ ***"CMD.EXE is not executable"***

In Windows NT, COMMAND.COM is called *CMD.EXE*. The *CMD.EXE* file has an invalid format and is not executable.

☞ ***"Division by 0 attempted"***

A division (or modulo) by 0 was attempted. The function was not executed.

☞ ***"End of command list encountered unexpectedly"***

The end of the command list was encountered unexpectedly. This error can occur when a closing brace ("]") is missing at the end of the command list.

☞ ***"File does not exist"***

The filename passed as a parameter could not be found in the specified path.

☞ ***"Help error: Unrecognized command"***

The command passed as a parameter is not a command internal to the interpreter, a command from the library, a user-defined command, or a command list name.

☛ ***"Illegal identifier in macro definition"***

The *MACRO* and/or *DELMACRO* commands were found in the macro definition. These two command words (and their first distinctive characters) cannot be part of a macro definition.

☛ ***"Illegal name for an identifier"***

The name specified as an identifier is not permitted. The first character of an identifier cannot be a digit.

☛ ***"Invalid number of parameters"***

The command was recognized, but the number of parameters for that specific command was incorrect.

☛ ***"Last command not recorded properly"***

The last command could not be properly saved to the record file.

☛ ***"Macro name already exists"***

The name passed as a parameter is already defined as a macro name.

☛ ***"Macro name does not exist"***

The macro name passed as a parameter does not exist.

☛ ***"Maximum length of string reached"***

The maximum of 1024 characters for a string has been reached.

☛ ***"Not enough memory to add a macro definition"***

There is not enough memory on the heap to dynamically allocate space for a new macro.

☛ ***"Not enough memory to create a command list"***

There is not enough memory on the heap to dynamically create a new command list.

☛ ***"Not enough memory to edit a command list"***

There is not enough memory to load the editor.

☞ ***"Not enough memory to execute another command list"***

There is not enough memory left on the heap to allocate memory to execute another command list.

☞ ***"Not enough memory to execute the command"***

Not enough memory is available to execute the command, the available memory has been corrupted, or an invalid block exists, indicating that the process making the call was not allocated properly.

☞ ***"Not enough memory to perform the initialization"***

There is enough memory to load the program, but not enough to enable it to perform all the dynamic initializations.

☞ ***"Not enough space left in editing buffer"***

There is not enough space left in the editing buffer to insert a new character, or to paste the contents of the paste buffer.

☞ ***"Out of memory: command list creation halted"***

There is not enough memory left to add another line to the command list. The creation of the command list is halted and you are returned to the interpreter prompt.

☞ ***"Parameter out of range"***

At least one of the parameters passed is out of range (greater than the maximum value or less than the minimum value) for that parameter type.

☞ ***"Parameters syntax error"***

Check for missing ")" or "]" in your parameter list and make sure the correct syntax has been used for all operator and constant values.

☞ ***"Parameters type mismatch"***

An attempt to pass a string parameter as a numerical value was detected, or any other type mismatch was detected.

☛ ***"Return index out of range"***

The specified return index is greater than the number of the returned value.

☛ ***"The argument list for the command is too big"***

The argument list that can be passed to command prompt is limited to 127 characters.

☛ ***"Too many subdefinitions. Macro not created"***

The definition of the macro, when checked against the other currently defined macros, caused an endless loop or too many subdefinitions occurred. The macro was not created, and the former definition (if there was any) has been lost.

☛ ***"Unrecognized command"***

The command is not a command internal to the interpreter, a command from the library, a user-defined command, or a command list name.

Note that, if you are using MIL, you will get this error message if you try to call a MIL processing function.

Interpreter Syntax

- **char** = any single representable character
- **bin_digit** = <0 or 1>
- **dec_digit** = <0 to 9>
- **hex_digit** = <0 to 9 and A to F>
- **letter** = (A to Z and a to z and "_")
- **num_const** = <<dec_digits> or
 <0D or 0d> <dec_digits> or
 <0B or 0b> <bin_digits> or
 <0X or 0x> <hex_digits>
- **char_const** = <' char '>
- **const** = <num_const or char_const>
- **string** = <" [char...]" or "[command]">
- **identifier** = <letters>
- **monadic_op** = <! or ~ or - or +>
- **dyadic_op** = <* or / or % or + or - or << or >> or
 == or < or > or != or <= or >= or
 & or | or ^ or && or || >
- **numerical_exp** = <numerical_const_exp> [<dyadic_op
 num_const_exp>...]
- **block** = <command or "{ command... }"> (See note)
- **parameter** = <numerical_exp or string or identifier or block>
- **command** = [num_const] <identifier> [parameter...]
- **filename** = string (for example, c:\intelcam.exe)
- **definition** = [char...]
- **comment** = <; [char...]>
- ❖ You must have only one command per line (CR is the command separator).

Appendix A: Interfacing a camera

This appendix will help you understand the descriptions and diagrams in your camera manual, and allow you to get your system up and running more quickly.

Overview

This appendix serves as an introduction to video and interfacing a camera to **Matrox** hardware. It will help you understand the descriptions and diagrams in your camera manual, and allow you to get your system up and running more quickly.

Depending on your level of knowledge, certain sections might be more useful to you than others:

- **Video Formats** identifies the various standard and non-standard video formats.
- **Standard analog video signal** describes how a standard analog video signal is produced, and also explains the various components of the video signal. This section also introduces some digitization-specific topics: *pixel clock* and *AC coupling and DC restoration*.
- **Color timing** describes basic concepts specific to a color video signal.
- **Non-standard video formats** describes some non-standard video formats.
- **Camera modes of operation** discusses the modes of operation for frame scan and line scan cameras.

Video formats

All video signals conform to a particular video format. The video format specifies such information as the type of video signal (analog or digital), synchronization signals, number of lines in an image, etc. There are standard and non-standard video formats.

Standard video formats

The standard monochrome video formats are RS-170 (used in the United States), RS-330 (used in Canada), RS-343 (used in Japan), and CCIR (used in Europe).

The standard color video formats are NTSC (United States, Canada, Japan, and parts of South America), PAL (Europe), and SECAM (France, Russia, and the republic states).

❖ *Appendix B* includes summaries of the characteristics of the RS-170, CCIR, NTSC, and PAL video formats.

Non-standard video formats

Non-standard video formats include negative-going video, digital video, and high resolution video.

- Negative-going video is an analog video signal where white or bright pixel data is represented by a more negative voltage than black or dark pixel.
- Digital video is a digitized waveform of any format of video signal (e.g., RS-170, NTSC, PAL), where the synchronization, blanking, and saturation levels have been assigned a digital value. (Synchronization, blanking, and saturation levels are described later.)
- High resolution video includes any camera with a spatial resolution of 1024 pixels x 1024 lines and higher. The difference between this video signal and standard video signals is the difference in the timing specifications and the signal period, along with the increased sampling rates required by the frame grabber.

Standard analog video signal

This section describes how a standard analog video signal is produced, and explains the various components of the video signal.

*How the video signal
is produced*

A video camera contains a two-dimensional area of photosensors (such as a charge-coupled device). These photosensors convert the energy of incident light particles into equivalent electrical charges. The charge of each sensor is then scanned out of the camera, in a left-to-right, top-to-bottom fashion, producing a continuous analog video signal that will eventually be digitized.

*Video timings and
DCFs*

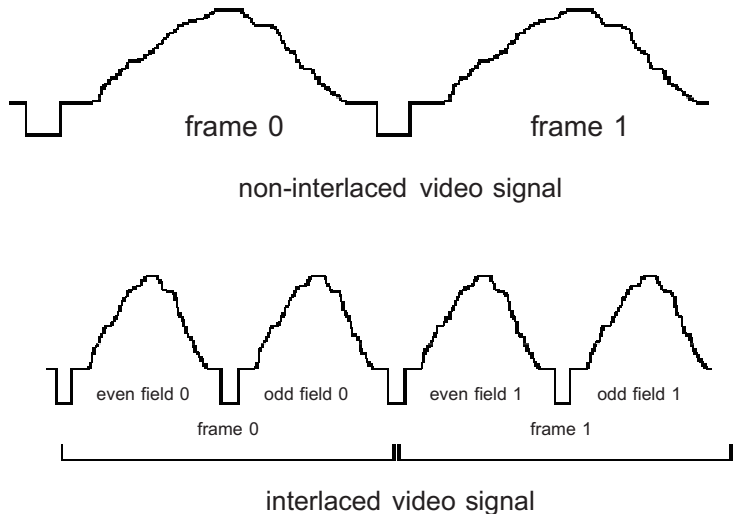
The respective widths of the sync pulse, the back porch, the active video period and the front porch are known as the video timings of the camera (each of these timings is defined later). These timings are required when creating a digitizer configuration format (DCF) file, using Matrox Intellicam, and can be found in the timing diagrams of your camera manual.

Interlaced and non-interlaced signals

In an *interlaced* video signal, the image frame is divided into two fields: an even field and an odd field. As such, the frame is read out of the sensor in an odd/even or even/odd fashion. Even fields consist of even-numbered lines and odd fields consist of odd-numbered lines.

In a *non-interlaced* video signal, the entire image frame is read out of the sensor using progressive scanning, that is, the entire frame is read out of the sensor at one time. The frame is not composed of separate fields.

The following illustrates interlaced and non-interlaced video signals:



Composite video signal

A *composite video signal* contains both timing (that is, synchronization) and pixel information in a single signal. The RS-170 monochrome video signal is an example of this type of video signal.

Synchronization

Synchronization indicates the end of a line or frame and the start of a new one. In composite signals, two types of pulses are needed to determine where every pixel should be "placed" in the final digital image. These pulses are the *horizontal sync* (hsync) and *vertical sync* (vsync).

A hsync pulse separates each video line and indicates where the beginning of the next scan line is to occur. A vsync pulse separates two frames (or fields) and indicates where the top of the next frame (or field) is.

During a horizontal or vertical sync, the video signal drops below the *blanking* level (explained later) to the *sync tip* level. For a RS-170 signal, the sync tip level is -0.286V.

Back and front porch

A vertical or horizontal blanking period is made up of a *front porch* period, a *back porch* period, and a sync pulse. The back porch period precedes the active video period, whereas the front porch period follows the active video period and precedes the next sync pulse.

Active video

The portion of the video signal above the black level contains the *active video* (that is, the part of the video waveform that is actually visible on the display screen), while the portion of the video signal below the black level contains all the synchronization information.

Amplitude and reference levels

A video signal has a definite voltage range that defines the *amplitude* of that signal. For example, since the RS-170 video signal ranges from -0.286V to $+0.714\text{V}$, it has an amplitude of 1V . The voltage of a black pixel is referred to as the *black level* or *reference black level*, and the voltage of a white pixel is referred to as the *saturation level* or *white reference level*. In the RS-170 video signal, the black level is $+0.054\text{V}$ and the saturation level is $+0.714\text{V}$.

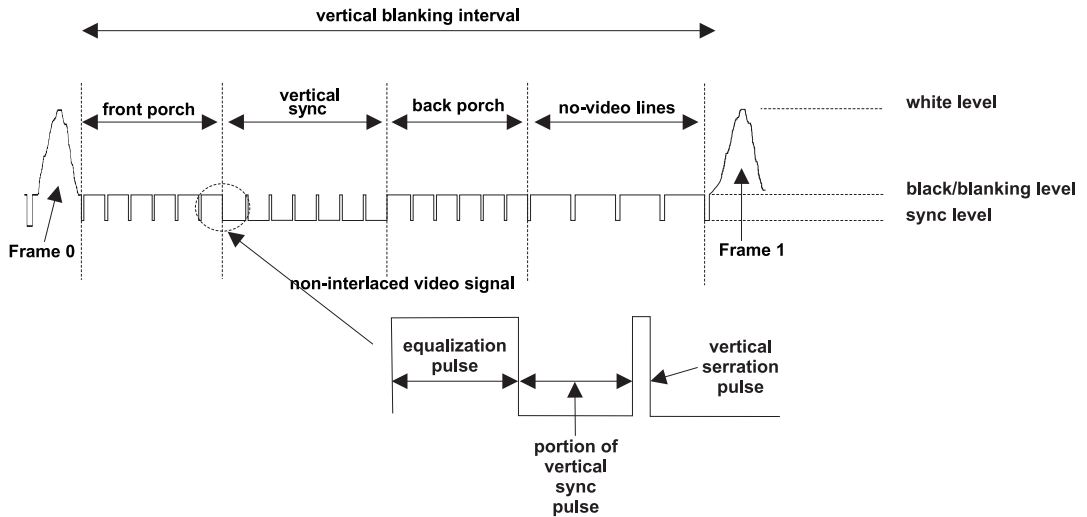
Blanking periods

A video signal has both *vertical* and *horizontal blanking* periods. The blanking period is the portion of a video signal after the end of a frame (*vertical blanking*) or line (*horizontal blanking*), and before the beginning of a new frame or line. During the blanking period, the video signal is "blanked" so that the path of the scan beam cannot be seen while it returns to the beginning of the next frame or line. To blank the video signal, the voltage is brought down to a *blanking voltage*, that is, equal to or below the black level.

- ❖ The video information during a blanking period does not contain any valid image data.

Vertical blanking

The vertical blanking interval occurs between two consecutive frames and consists of a front porch, a vsync pulse, a back porch, and a "no-video lines" period, as shown:



The front porch, vsync and back porch (as shown in the above figure) have been discussed previously. The "no-video lines" period is the part of the back porch that precedes the next frame of video information. The no-video lines period does not contain *serration pulses*.

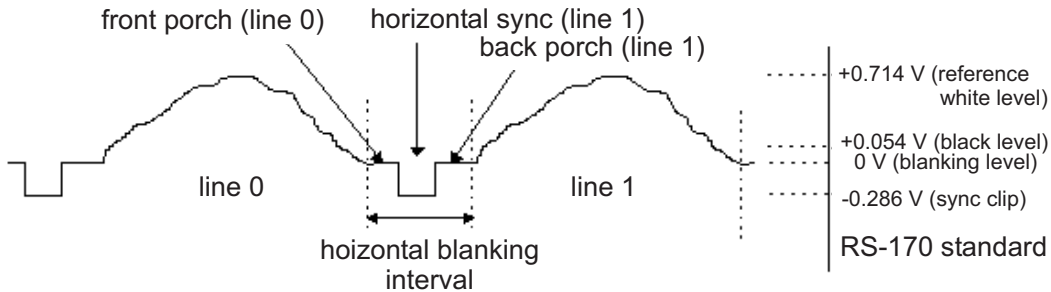
Serration pulses

Serration pulses, which occur during the vertical sync interval, are pulses used to synchronize video equipment. These pulses have a frequency equal to twice the normal horizontal scan rate.

❖ As shown above, the polarity of the sync pulses in the vsync period are inverted.

Horizontal blanking

The horizontal blanking interval occurs between two consecutive lines and consists of the front porch of the previous line, a hsync pulse, and the back porch of the current line, as shown:



Clamping

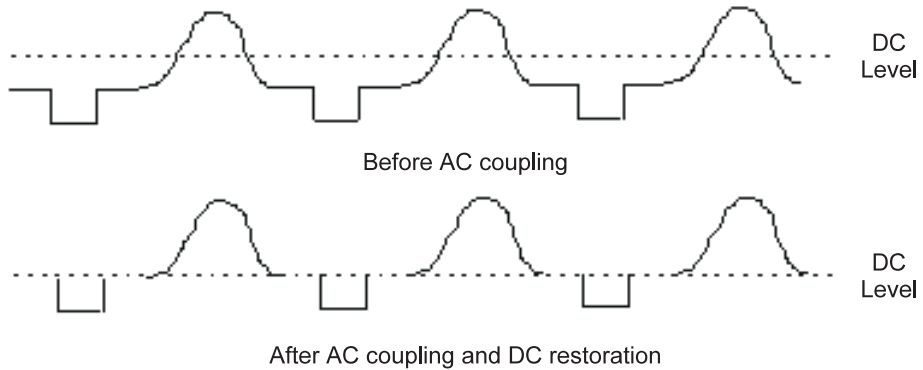
DC restoring of the signal, called *clamping*, usually occurs during the back porch of the hsync interval, though in some cameras it may occur during the sync pulse or even the front porch.

The *clamping period* is the period during which the reference voltage level (or DC offset) of an analog signal is determined.

If the video output is DC-coupled the location of the actual image information is known. Therefore, the signal can be properly digitized since a pixel's brightness can be determined by comparing its voltage against a fixed voltage reference.

If the DC offset of a video signal is not known, the location of the actual image information is not known. Therefore, the DC offset is removed using a process known as *AC coupling*. Then, a known DC offset is added to the video signal using a process known as *DC restoration* or *clamping*. *DC restoration* or *clamping* uses DC line clamps to set the signal's black reference signal level in the analog-to-digital (A/D) converter to a stable level and a known starting point. This defines a

reference point and, as such, the video signal can be properly digitized. Each pixel's brightness can be determined by comparing its voltage against a fixed voltage reference. The effects of AC coupling and DC restoration are illustrated below:



Pixel clock

To produce the digital image, you must specify a sampling rate, called the *pixel clock*, which determines how many pixels will be extracted from the active portion of the analog signal. In other words, the pixel clock is used to divide the incoming horizontal line of video into separate pixels by specifying the exact location, in time, of each pixel.

In general, the pixel clock timing comes from one of three timing mechanisms: a crystal oscillator, a phase-locked loop (PLL), or an external pixel clock.

- A crystal oscillator provides an accurate and stable frequency at standard rates.
- A phase-locked loop (PLL) creates a pixel clock by locking itself to the hsync of the video signal. This ensures that the number of pixels in each line remains constant. Depending on the design of the PLL, a certain degree of error, called *pixel*

jitter, occurs. Pixel jitter is measured in nanoseconds. If the value of the jitter is an appreciable proportion of a single pixel time, the quality of the captured data will be reduced.

- A pixel clock can be generated by the digitizer and fed to the camera, or generated by the camera and fed to the digitizer. In general, an external pixel clock can support very high digitization rates and allows for exact pixel location.

Clock exchange

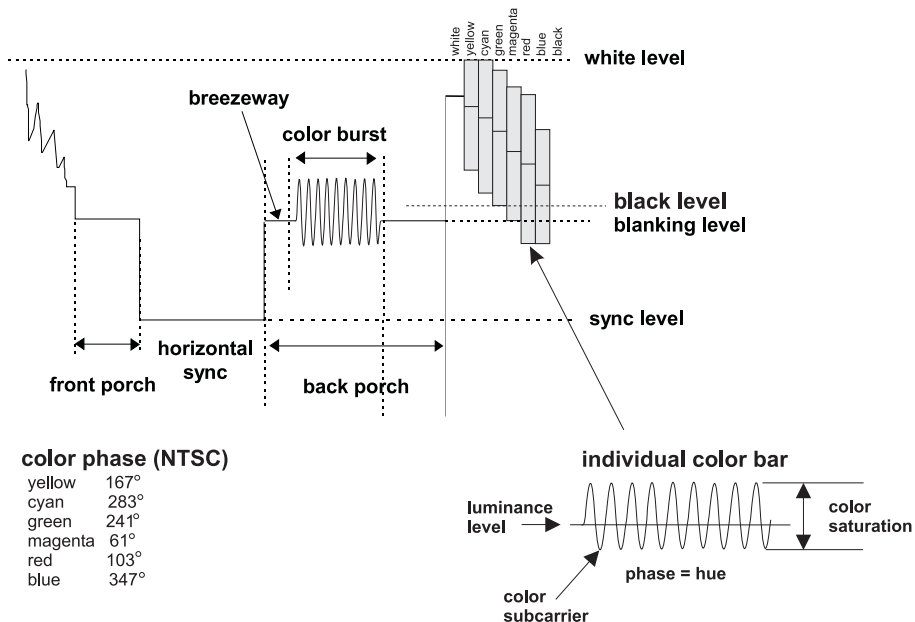
As a general rule, if the camera has a pixel clock input, a pixel clock can be supplied to it. Sometimes, the internal workings of a camera dictate that, from the pixel clock sent by the frame grabber, the camera must generate and send back a strobe of a different frequency that corresponds to the rate at which data is being read out. This is called *clock exchange*.

Color timing

The video signals for color formats are, in general, modifications of the monochrome formats. For example, the NTSC format is simply a modified RS-170 video signal that includes color information. Similarly, the PAL format is a modified CCIR video signal that includes color information.

In general, a color format consists of black-and-white information (*luminance*) and color information (*chrominance* or *color subcarrier*). The color subcarrier's *amplitude* represents the saturation and the *phase angle* represents the hue.

In addition to the luminance and chrominance signals, an additional signal that contains information on how to decode the colors in each video line is needed. This signal, known as the *color burst*, occurs after the horizontal sync, during the back porch. The period preceding the color burst is known as the *breezeway*, as shown:



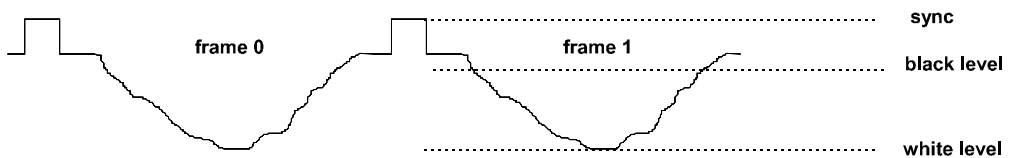
If the luminance and chrominance exist on a single signal, the signal is a composite color signal. To decode a composite color signal, the chrominance and luminance first have to be separated. The chrominance can be isolated using a *chroma bandpass* filter, or the luminance can be isolated using a *chroma trap*. Then, the colors can be properly decoded for output, using a *chroma demodulator*.

Non-standard video formats

Non-standard video formats include negative-going video, digital video, and high resolution video.

Negative-going video

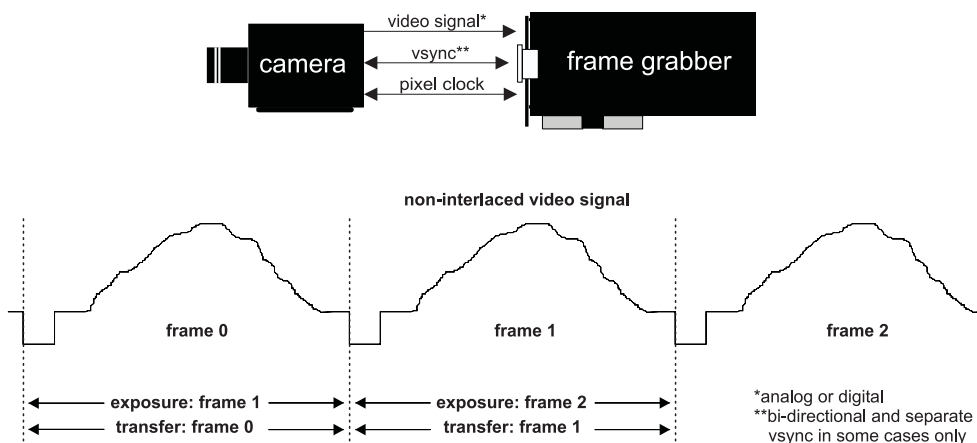
Negative-going video is an analog video signal where white or bright pixel data is represented by a more negative than black or dark pixel. In general, a negative-going video signal can be represented as follows:



Digital video

Digital composite video is essentially the same waveform as the analog composite RS-170 video signal, except that the waveform is digital and the data-carrying signals are restricted to one of two voltage levels: logic 1 or 0. Representing data in digital form is useful since the waveform can be regenerated with minimum noise and distortion as it is being transferred.

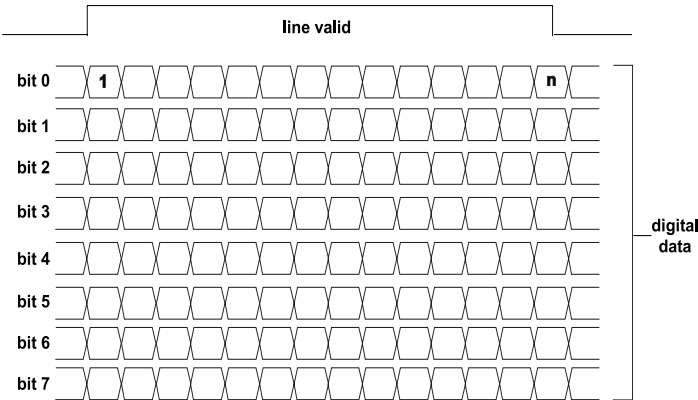
The following shows an analog composite video waveform and a digital composite video waveform:



In digital video, pixel values are represented by an n -bit system, where a value between 0 and 2^n represents the brightness value. For example, in an 8-bit system, pixels can have 256 possible values, ranging from 0 to 255. As n increases, image information increases. For monochrome images, as n increases,

more shades of gray are available, which results in a more accurate representation of the grabbed image. The following illustrates 8-bit digital data:

Each pixel is defined by the sampling of a single line at a certain point in time. The pixel can be represented with 256 brightness values, as in this 8-bit example ($2^8=256$).



Digital video data is usually transmitted on a pixel-by-pixel basis in the form of several bits in parallel. Each bit is transmitted on an individual signal line (using the TTL logic levels standard) or on a pair of signal lines (using the differential RS-422 standard).

TTL and RS-422

TTL is a medium-to-high-speed family of logic-integrated circuits, while RS-422 is a medium-range, differential-signaling-pair standard. With RS-422, digital information can travel over a longer distance without the introduction of as much noise as with TTL.

Camera modes of operation

Camera terminology varies from one manufacturer to another, so the definitions found here are as Matrox uses them.

Note that "internal" refers to the camera end and "external" refers to the board end. In addition, connections mentioned in the following sections are general ones. Particular cameras might require additional connections for auxiliary control signals, etc. All required connections are specified in the camera manual.

Typically, cameras can be operated in any one of several modes, as follows.

Frame scan cameras

Frame scan cameras can be operated in any of the following modes: continuous mode, pseudo-continuous mode, trigger mode, asynchronous reset mode, control mode, and long exposure (integration) mode.

Continuous mode

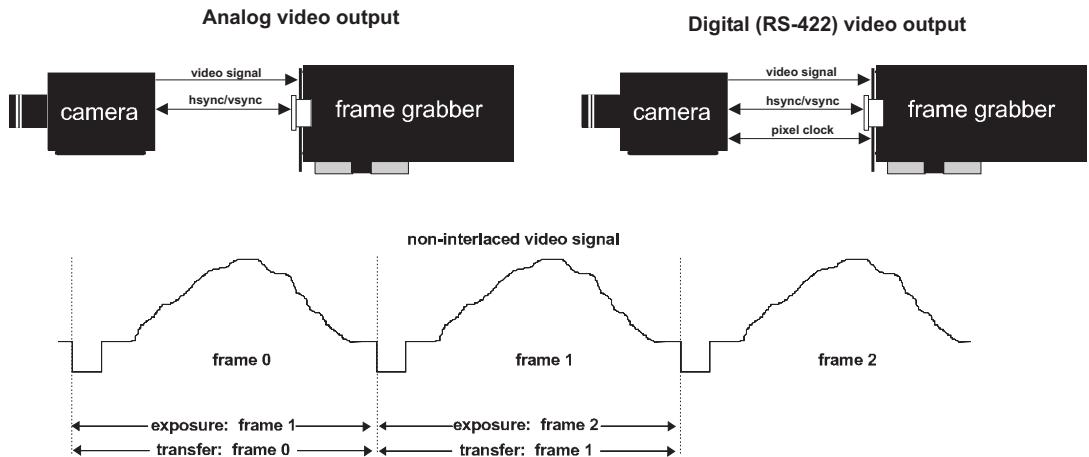
In continuous mode, the camera continuously outputs images at a fixed frame rate, usually 30 frames or 60 fields per second (North American timings) or 25 frames or 50 fields per second (European timings). In general, the exposure time is the reciprocal of the frame rate. By adjusting the camera, you might be able to use a shorter exposure time. The frame rate, however, does not change. In continuous mode, exposure of the current frame and transfer of the previous frame occur concurrently. Therefore, in this mode, exposure time cannot exceed the reciprocal of the frame rate.

If the camera outputs an analog video signal where both the hsync and vsync are combined with video data (composite video signal), then that signal alone is required by the frame grabber to operate in continuous mode. Some cameras can output an analog video signal where only the hsync is composite, although this is not typical. In such a case, a separate digital vsync signal (e.g., a frame enable or a trigger signal) is supplied by the camera to the frame grabber or vice-versa.

❖ Separate digital syncs can also be used when the camera outputs a fully composite analog signal.

If the camera outputs a digital video signal, both the hsync and vsync are usually separate digital signals provided by the camera or supplied by the frame grabber. Some cameras combine the hsync and vsync to form a single digital composite sync. Finally, a pixel clock may be provided by the camera or supplied by the frame grabber, if required. They can be supplied by both in the case of clock exchange (see the *Clock exchange* section).

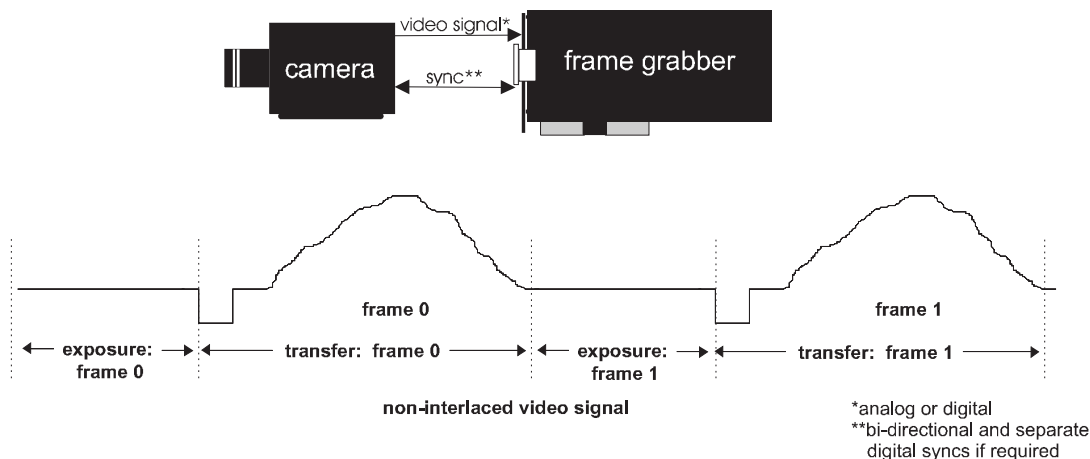
The following illustrates analog and digital output, and timings in continuous mode:



Pseudo-continuous mode

In pseudo-continuous mode, the camera continuously outputs images at a frame rate that is determined by the exposure time and the frame transfer time.

The exposure time might be selectable by adjusting the camera; however, the frame transfer time is fixed and is a characteristic of the camera. As can be seen in the timing diagrams in the manual of any camera that operates in this mode, exposure and transfer of a frame occur sequentially, and exposure of a new frame only starts once the previous frame has been fully transferred. As such, the frame rate is the reciprocal of the sum of the exposure time and the frame transfer time. The camera sets an upper limit on the exposure time but, as opposed to continuous mode, the exposure time can be much longer than the frame transfer time. The signals involved in this mode are the video output and syncs. As with continuous mode, these signals may be combined with video data (composite) or separate digital syncs can be used. The following illustrates the timings for pseudo-continuous mode.



Trigger mode

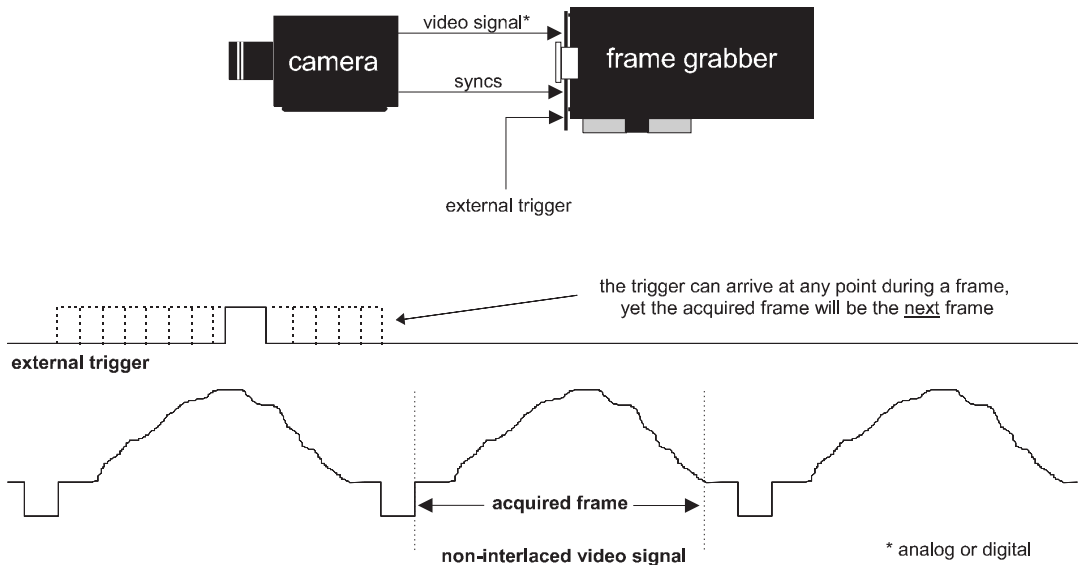
The trigger mode is used to capture a single image or a sequence of images.

In trigger mode, as in continuous mode, the camera continuously outputs images at a fixed frame rate. However, to grab a frame, the frame grabber must receive an external trigger signal.

Note that the frame grabber will ignore an external trigger pulse that arrives before the current frame period is over. Therefore, to ensure that all required frames are captured, the shortest time between external trigger pulses must be greater than the sum of the exposure time and the frame transfer time.

In addition to the external trigger signal, the video output and syncs are provided to the frame grabber.

The following illustrates the timings for trigger mode:



Asynchronous reset mode

To grab a frame in asynchronous reset mode, either an external trigger signal is provided to or an internal trigger is generated by the frame grabber. An internal trigger can be periodic or aperiodic (controlled by software). The frame grabber, in turn, triggers the asynchronously resettable camera to initiate exposure. The trigger signal from the frame grabber to the camera is referred to as the exposure signal. The camera is resynchronized on the exposure pulse. The delay from the time the frame grabber is triggered to the time it starts exposing is programmable.

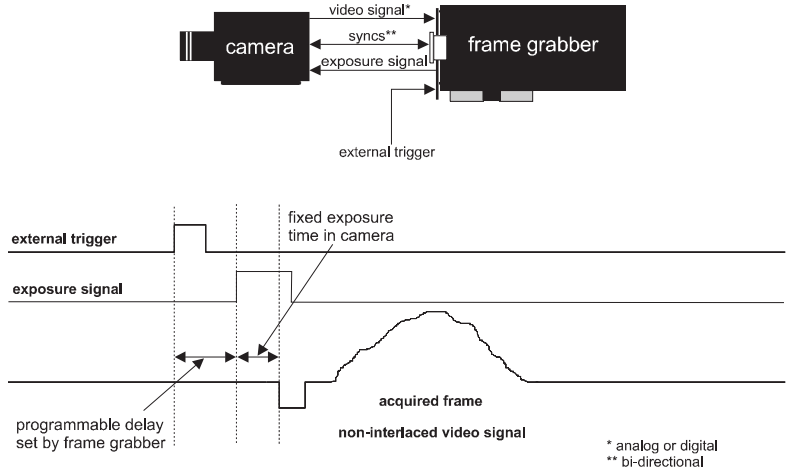
There are three types of asynchronous reset modes used by cameras, as follows:

- **Vertically asynchronously resettable:** The vertical timings are reset on the exposure pulse.
- **Vertically and horizontally asynchronously resettable:** Both the vertical and horizontal timings are reset on the exposure pulse.
- **Fully asynchronously resettable:** The vertical and horizontal timings and the pixel clock are reset on the exposure pulse.
 - ❖ Examine the timing diagrams found in your camera's manual to determine which of the three cases corresponds to your particular camera's asynchronous reset mode.

In asynchronous reset mode, the exposure time is adjusted on the camera itself. Some cameras will ignore an exposure pulse that arrives before the current frame period is over, while others will resynchronize on this new pulse, discarding all current information. In general, to avoid losing information, the shortest time between external trigger pulses should be greater than the sum of the exposure pulse width, the exposure time, and the frame transfer time.

The signals used in asynchronous reset mode are an external trigger signal provided to the frame grabber, an exposure signal supplied from the frame grabber to the camera, and the video output (analog or digital) and syncs.

The following illustrates the timings for asynchronous reset mode:



Control mode

In control mode, the exposure time is controlled externally, using a frame grabber. Usually, the camera is triggered asynchronously. The asynchronous reset pulse is provided by the frame grabber, to which an external trigger source is connected. The asynchronous reset pulse is referred to as the exposure signal.

In this mode, the camera is resynchronized on the exposure signal. The width of the exposure signal determines the exposure time.

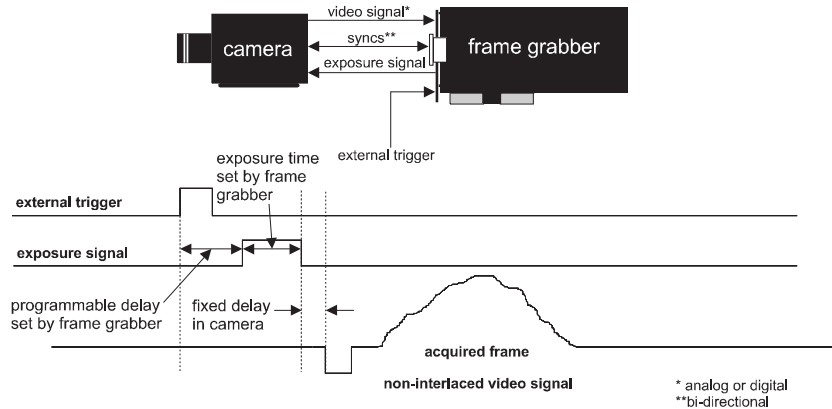
❖ You can adjust the width of the exposure signal on the frame grabber, using Intellicam.

Some cameras will ignore an exposure signal that arrives before the current frame period is over, while others will resynchronize on this new pulse, discarding all current information. To avoid losing information, the shortest time between external trigger pulses should be greater than the sum of the exposure signal width and the frame transfer time.

The signals used in this mode are an external trigger signal provided to the frame grabber, an exposure signal supplied by the frame grabber to the camera, and the video output and syncs.

- ❖ Use the control mode when control over the start and exposure time of an image is required.

The following illustrates the timings for control mode:



Long exposure or integration mode

In long exposure or integration mode, the exposure time is controlled internally via switches on the camera, or externally by way of the frame grabber. In this mode, an external trigger signal is provided to the frame grabber which, in turn, triggers the camera. In this mode, the trigger signal from the frame grabber to the camera is referred to as the exposure signal.

With most cameras, the exposure pulse is latched to the hsync. The camera uses this hsync to initiate frame transfer on its next vsync. In general, the exposure time must be specified as a multiple of fields or frames, where one frame time (the frame transfer time) is equal to the reciprocal of the frame rate of the camera when operated in continuous mode. Note that one field time is half of one frame time.

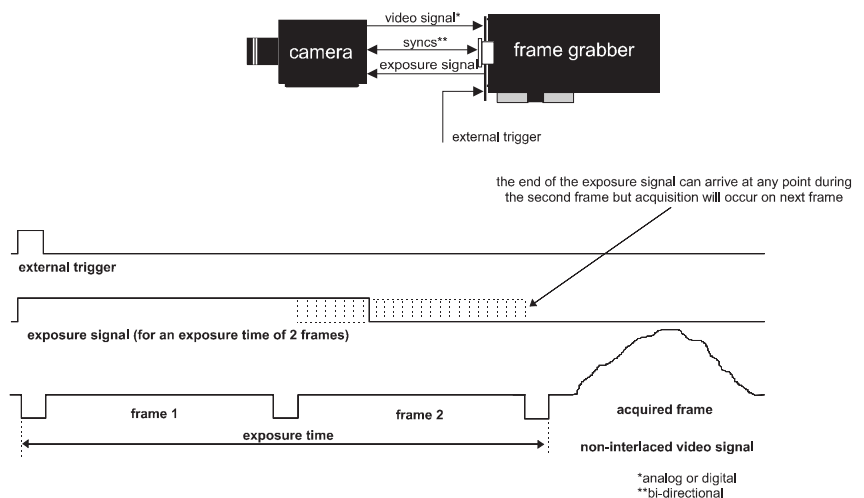
- ❖ Use long exposure or integration mode when an exposure time greater than one frame time is required. Most cameras will ignore the end of an exposure pulse that arrives before the current frame period is over, while others will latch to the pulse and initiate the next exposure directly afterward.

To ensure capturing an image:

- If the exposure is *internally controlled*, the shortest time between external trigger pulses should be greater than the sum of the exposure pulse width, the exposure time, and the frame transfer time.
- If the exposure is *externally controlled*, the shortest time between external trigger pulses should be greater than the sum of the exposure pulse width and the frame transfer time.
- ❖ The width of the exposure pulse determines the exposure time and is adjusted on the frame grabber using Intellicam.

The signals used in this mode are an external trigger signal provided to the frame grabber, an exposure signal supplied by the frame grabber to the camera, and the video output (analog and digital) and syncs.

The following illustrates the timings for long exposure or integration mode:



Line scan cameras

Line scan cameras can be operated in one of the following modes: continuous line scan rate mode, variable line scan rate mode, or variable line scan rate with triggered frame mode.

Continuous line scan rate mode

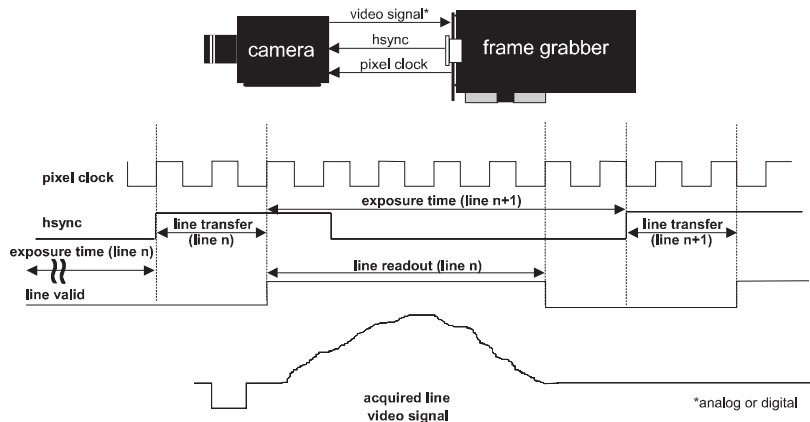
In general, in continuous line scan rate mode, the frame grabber supplies an exposure signal to the line scan camera. Sometimes, the hsync is used instead of the exposure signal. The frequency of this hsync signal determines the line scan rate. Line transfer is initiated with every hsync. and is followed by line readout.

Unless the camera features exposure control, the exposure time is the reciprocal of the line scan rate.

A pixel clock is supplied by the frame grabber to the camera. Some cameras also return another clock (strobe) to the frame grabber, that is, a clock that is derived from the first clock, to be used as the real pixel clock (clock exchange).

The signals used in fixed line scan rate mode are a pixel clock and hsync (both supplied by the frame grabber to the camera), a returned strobe signal (with some cameras), and a video output that can be analog or digital.

The following illustrates the timings for fixed line scan rate mode:



Variable line scan rate mode

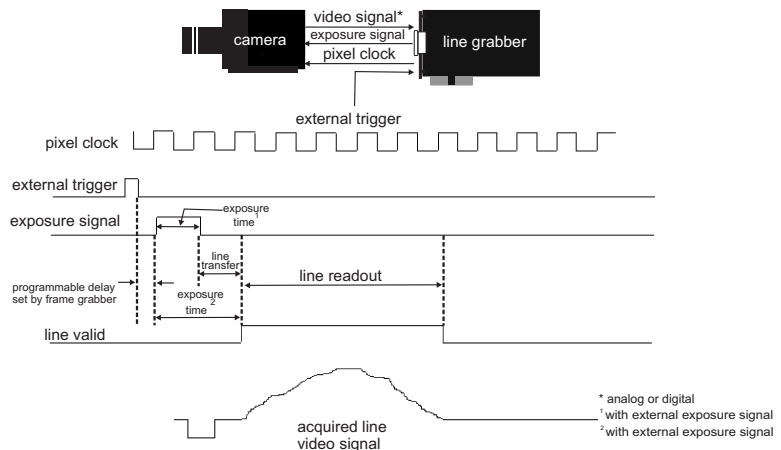
In variable line scan rate mode, an external trigger signal is provided to the frame grabber which, in turn, triggers the camera to initiate line readout. In this mode, the trigger from the frame grabber to the camera is called the exposure signal.

The time interval between external trigger pulses determines the line scan rate. This time interval must be greater than the sum of the exposure time and the line transfer time. With external exposure control, the length of the exposure signal determines the exposure time. Without exposure control, exposure time is the reciprocal of the line scan rate. With internal exposure control, the exposure time is set from the camera (using switches or control bits).

A pixel clock is usually supplied to the camera by the frame grabber. Some cameras also return another clock (strobe) to the frame grabber, that is, a clock that is derived from the first clock, to be used as the real pixel clock (clock exchange).

The signals used in this mode are an external trigger signal (provided to the frame grabber), a pixel clock and exposure signal (both supplied by the frame grabber to the camera), a returned strobe signal (with some cameras), and a video output that can be analog or digital. The camera can also return a line valid signal.

The following illustrates the timings for variable line scan rate mode:



Variable line scan rate with triggered frame mode

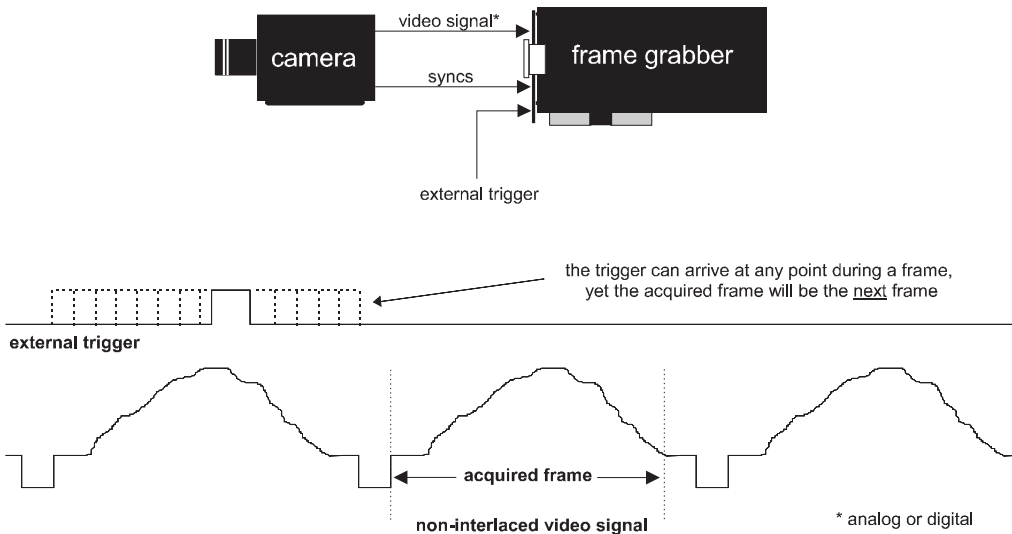
In variable line scan rate with triggered frame mode, the frame grabber receives two external triggers: a line trigger and a frame trigger. The line trigger is continuous; however, it has a variable rate. This line trigger triggers the camera to start reading out a line. When the frame trigger occurs, a specified number of lines are acquired. In this mode, a trigger from the frame grabber to the camera is called the exposure signal.

A pixel clock is usually supplied to the camera by the frame grabber.

The signals used in this mode are external line and frame trigger signals, both provided to the frame grabber; a pixel clock and exposure signal, both supplied by the frame grabber to the camera; a returned strobe signal (with some cameras); and a video output, which can be analog or digital.

❖ This is actually a submode of variable line scan rate mode, and can be used with Matrox boards, including Image-Series and Genesis.

The following illustrates the timings for variable line scan rate with triggered frame mode:



Summary of camera modes

The following tables summarize the various camera modes for frame scan and line scan cameras. Note that "internal" refers to the camera end and "external" refers to the board end.

Frame scan cameras

Camera Modes	Connections
Continuous mode: <ul style="list-style-type: none"> • continuous video • internal exposure control • exposure time cannot exceed frame transfer time • fixed frame rate is independent of exposure time 	<ul style="list-style-type: none"> • video and sync signals between camera and frame grabber (syncs can be provided by the frame grabber)
Pseudo-continuous mode: <ul style="list-style-type: none"> • continuous video • internal exposure control • exposure time can be much longer than frame transfer time • frame rate is a function of exposure time 	<ul style="list-style-type: none"> • video and sync signals between camera and frame grabber
Trigger mode: <ul style="list-style-type: none"> • internal exposure control • external trigger 	<ul style="list-style-type: none"> • video and sync signals connected between camera and frame grabber • external trigger signal connected to frame grabber
Asynchronous reset mode: <ul style="list-style-type: none"> • internal exposure control • external trigger 	<ul style="list-style-type: none"> • video, sync and exposure (frame grabber acting has asynchronous reset) signals connected between camera and frame grabber • external trigger signal connected to frame grabber
Control mode: <ul style="list-style-type: none"> • external exposure control • external trigger 	<ul style="list-style-type: none"> • video, sync and exposure (frame grabber acting has asynchronous reset plus actual exposure) signals connected between camera and frame grabber • external trigger signal connected to frame grabber
Long exposure or integration mode: <ul style="list-style-type: none"> • internal or external exposure control • exposure times longer than one frame • external trigger 	<ul style="list-style-type: none"> • video, sync and exposure (trigger) signals connected between camera and frame grabber • external trigger signal connected to frame grabber

Line scan cameras

Camera Modes	Connections
Continuous line scan rate mode: <ul style="list-style-type: none"> line scan rate determined by the frequency of the hsync signal 	<ul style="list-style-type: none"> video and sync signals between camera and frame grabber (syncs can be provided by the frame grabber)
Variable line scan rate mode: <ul style="list-style-type: none"> line scan rate determined by time between external trigger pulses internal or external exposure time control 	<ul style="list-style-type: none"> video, sync and exposure (trigger) signals connected between camera and frame grabber external trigger signal connected to frame grabber
Variable line scan rate with triggered frame mode: <ul style="list-style-type: none"> line scan rate determined by time between external trigger pulses internal or external exposure time control 	<ul style="list-style-type: none"> video, sync and exposure (trigger) signals connected between camera and frame grabber external line and frame trigger signals connected to frame grabber

Appendix B: Glossary

This appendix provides the definitions of terms used in this manual and in the interface.

Glossary of terms

■ **active period**

This is the state following the back porch period. This state contains the valid video information (pixels or lines) generated by the camera. During this state, the digitizer grabs the video information and places it into memory.

■ **active video**

Related to the video signal, the active video period is the interval in which the signal actually carries valid pixel data or information. For more information, refer to the **video timings** definition.

■ **asynchronous reset trigger mode**

This is a mode in which the camera can be externally reset to start a new frame. The camera resets its vertical timing. Depending on the model, it can also reset its horizontal timings and even resynchronize its pixel clock. In other words, the reset is vertically asynchronous and may be horizontally or totally asynchronous.

■ **back porch**

This is the state following the sync period and preceding the active video period. During this state, the video information is not valid.

Note that, with some cameras, it might be difficult to find the duration of the sync and back porch because the camera specification describes only the timing between the beginning of the line/field and the first valid pixel/line. For more information, refer to the **video timings** definition.

■ **band**

One of the color components of a buffer. A grayscale image requires just one band. A color image requires three bands, one for each color component.

■ **black voltage level**

(The following applies to analog video). The black voltage level is the voltage corresponding to a black pixel. Any voltage level below the black voltage level (such as the voltage levels used for synchronization signals) is referred to as "blacker than black".

■ **blanking period**

The blanking period is the portion of a video signal after the end of a line, and before the beginning of a new one. During this period, where displays are concerned, the video signal is "blanked" so that a scan line can be brought back to the beginning of a new line. To do so, the voltage is set to a blanking voltage. The blanking voltage might be the same as the black voltage level.

■ **bloc synchronization**

Bloc synchronization refers to the synchronization information provided in the RS-343 standard (which is a modified version of the RS-170 standard). This type of synchronization signal does not have serration or equalization pulses during the vertical retrace. In addition, horizontal synchronization signals do not occur during the vertical synchronization period.

■ **camera type**

See **frame scan camera type** and **line scan camera type**.

■ **child area**

A child area refers to a specific portion of an image (that is, a specific region of interest). Any change made to the child area affects the entire image. In other words, an operation performed on the child area is also performed on the corresponding region of the parent (full) image. However, the child area is an image in its own right. Whenever the entire image can be used, you can use the child area instead to affect only a part of the entire image. When a child area is specified, results are returned relative to the child area's coordinates rather than the parent image's coordinates.

- **clamping period**

The clamping period is the location where the reference voltage level of an analog signal is determined. It is found in the inactive video region (back porch, front porch or sync pulse), and is characterized by a constant video voltage level that defines the black level of the video signal.

- **composite input signal**

A composite input signal refers to a signal that carries both the horizontal and vertical synchronization information, along with the video information.

- **composite sync**

Composite sync refers to a synchronization signal made up of two components: one vertical and one horizontal.

- **DCF**

A DCF (Digitizer Configuration Format) is a document, generally stores in a file, that is created by Matrox Intellicam. This document contains the register values with which to configure your Matrox digitizer. It is used by Matrox software to initialize the digitizer such that it accepts the particular video signals from a specific camera.

Such files have a *.dcf* extension.

- **digitizer configuration format document**

See **DCF**.

- **dual-tap camera**

This one-dimensional scan-type camera grabs on a line basis with constant or variable horizontal timings. Sometimes called a **line scan camera**.

■ **electrical format - TTL and RS- 422**

Electrical format pertains to the type of signal generated by a digital source, TTL or RS-422. Each format defines a specific voltage and electrical characteristics.

- The TTL format signal is characterized by voltage levels of 0V and 5V that correspond to the standard logical states 0 and 1, respectively.
- The RS-422 format signal is composed of two complementary signals (differential), SIGNAL+ and SIGNAL -, from which we can extract the standard logical states 0 and 1.

■ **equalization pulses**

Equalization pulses, like serration pulses, are sync pulses used to synchronize video equipment. These small pulses might or might not occur in the composite sync (csync) signal, depending on the type of camera used. They appear during the vertical blanking period, at the end of the field, and following the serration pulse.

■ **exposure**

Exposure refers to the period during which the image sensor of a camera is exposed to light. As the length of this period increases, so does the image brightness.

■ **exposure clock**

The exposure clock specifies the frequency with which the exposure signal is calculated.

■ **exposure signal**

An exposure signal refers to the signal generated by a timer.

■ **field**

(The following applies to frame scan cameras). A field refers to one half of a frame in interlaced modes. There is an odd and an even field for each frame.

- **frame**

(The following applies to frame scan cameras). A frame refers to an entire image.

- **frame scan camera type (i.e., periodic)**

This two-dimensional camera is generally a two-dimensional camera that has continuous scan, with constant horizontal and vertical timings. Examples of frame scan cameras include any RS-170A or CCIR timing compatible scan camera.

Some frame scan cameras have variable scan, typically triggered by an external signal (vsync, trigger, etc.) More specifically, they have a constant horizontal timing and a variable or constant vertical timing, for example, a camera that generates a series of frames separated by a variable delay. (This delay might be an exposure time, variable blanking, the delay of a part sensor trigger, etc.).

- **front porch**

This is the last state in a line/field. It follows the active video period and precedes the state of the next line/field. The video information is not valid during this state.

Note that, based on your camera type, it's possible that one or more states are not present in the camera video timings. If a specific state is missing, set its corresponding timing value to zero, or the minimum allowed by your frame grabber. For more information, refer to the **video timings** definition.

The above also applies to **back porch**.

- **horizontal sync**

Horizontal sync refers to the horizontal reference signal. It is used to horizontally synchronize video devices to each other (e.g., a camera and a frame grabber).

■ horizontal video timings

Horizontal video timings refer to the portion of the camera signal associated with the lines in a frame.

■ input-signal gain

(The following applies to analog video). The input-signal gain is a factor by which to electrically amplify the incoming video signal.

■ interlaced

(The following applies to frame scan cameras). Interlaced describes a camera that sends a frame in two distinctive parts: one that consists of the even-numbered lines, and another that consists of the odd-numbered lines. These parts are known as the even field of the frame, and the odd field of the frame. The lines of the frame are written to the buffer in an interleaved fashion (that is, the odd and even lines are properly intermixed).

■ interpreter start-up file

When MILINTER starts up (which occurs when Intellicam starts up, an interpreter start-up file (*milinter.ini*) is executed to initialize the interpreter state and the pre-defined macros. This start-up file is, in essence, a command list file that can include any MILINTER command or call other command lists.

You can change the contents of this file and/or the name of this interpreter start-up file, if required.

■ keying

Keying (or digital keying) is an effect that makes portions of a frame buffer (foreground or overlay frame buffer) transparent so that corresponding areas of an underlying frame buffer can be seen through it. Transparency occurs when foreground frame buffer pixels are of a predetermined color (keying color). In this case, the pixels at these corresponding locations in the underlying frame buffer are displayed instead.

If keying (overlay) is available on your image processing board or frame grabber, Intellicam displays its Windows interface from the foreground frame buffer (typically, the VGA frame buffer), and displays its images from your board's underlying frame buffer. Otherwise, it displays both from the same frame buffer (typically, the VGA frame buffer).

- **line scan camera type**

This one-dimensional scan type camera grabs on a line basis with constant or variable horizontal timings. Sometimes called a **dual-tap camera**.

- **multi-tap camera**

A multi-tap camera is a camera that has two or more outputs. Therefore, two or more pixels can be transmitted simultaneously. This setup allows for faster frame/line rates without increasing the pixel clock.

- **non-interlaced**

(The following applies to frame scan cameras).

Non-interlaced describes a camera that sends a single signal per frame. The lines of the frame are written into a buffer in a non-interleaved fashion (that is, sequentially).

See also: **progressive scan**.

- **pedestal**

Related to an analog video signal, the pedestal is defined as the DC voltage offset between the blanking and the black voltage level.

- **pixel clock**

The pixel clock is a signal (usually digital) that specifies the exact location in time of every active or blank pixel.

The pixel clock frequency (or sampling rate) determines how many pixels of digital data will be extracted from the active portion of the analog or digital signal.

■ polarity

Polarity indicates the active edge of a trigger or synchronization signal. A positive edge trigger signal means that the rising edge is the active edge, while a negative edge means that the falling edge is the active edge, whereas a falling signal activates a negative edge trigger signal.

■ progressive scan

(The following applies to frame scan cameras). Progressive scan describes a camera that sends a single signal per frame. The lines of the frame are written into a buffer in a non-interleaved fashion (that is, sequentially).

See also: **non-interlaced**.

■ reference levels

(The following applies to analog video). When digitizing images, the black and white reference levels determine the zero and full-scale values, respectively, of the input voltage range. The analog-to-digital converters convert any voltage above the reference level to the specified maximum white pixel value and any voltages below the black reference level to a zero pixel value.

■ serration pulses

Serration pulses, like equalization pulses, are synchronization pulses used to synchronize video equipment. These small pulses might or might not occur in the composite sync (csync) signal, depending on the type of camera used. They appear during the vertical blanking period, directly at the beginning of the field/frame.

■ synchronization signal

A synchronization signal is a signal that indicates the end of a line or frame and the start of a new one (horizontal sync signal or vertical sync signal).

■ sync pulse period

This is the state that indicates the beginning of a line (horizontal sync) or the beginning of a field/frame (vertical sync). During this state, the video information is not valid.

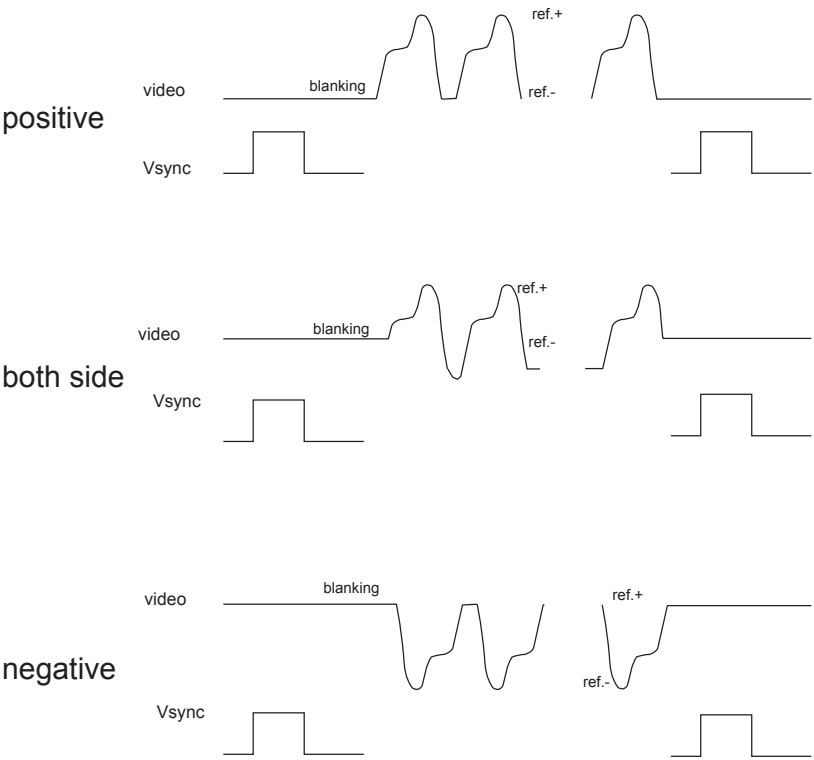
■ **system**

In general, in Intellicam, a system refers to a board with video acquisition capabilities that you have installed on your computer.

■ **swing type**

The swing type of an analog video signal describes the relationship between the voltage levels of the active video and the vertical blanking. In a positive swing signal, the voltage of the active video is above (more positive than) the blanking level. In a negative swing signal, the active video signal is below (more negative than) the blanking level. An active-video voltage swing that is both above and below the blanking level is defined as a "both side" swing-type signal.

The following diagram shows the three different types of swing-type analog video signals.



■ timer

A timer is an electronic circuit that measures time. In general, a timer is used to generate an exposure signal at the beginning of a grab. When you use a timer as a trigger, a frame is grabbed a specified amount of time after the timer is started. This is known as the exposure signal. The exposure signal remains active for a specified amount of time (known as the exposure time or pulse width).

■ video format

Below are the characteristics of the various video formats.

- | | |
|---------------|---|
| RS-170 | <ul style="list-style-type: none"> ■ North American monochrome video format standard ■ 640 pixels x 480 lines ■ analog, interlaced ■ 525 lines per frame, 485 active lines per frame ■ line time: 63.556 μs, active line time: 52.66 μs ■ line scan frequency: 15.734 kHz |
| CCIR | <ul style="list-style-type: none"> ■ European monochrome video format standard ■ 768 pixels x 572 lines ■ analog, interlaced ■ 625 lines per frame, 575 active lines per frame ■ line time: 64 μs, active line time: 52 μs ■ line scan frequency: 15.62 kHz |
| NTSC | <ul style="list-style-type: none"> ■ North American color video format standard ■ 640 pixels x 480 lines ■ analog, interlaced ■ 525 lines per frame, 485 active lines per frame ■ line time: 63.556 μs, active line time: 52.66 μs ■ line scan frequency: 15.734 kHz |

PAL

- European color video format standard
- 768 pixels x 572 lines
- analog, interlaced
- 625 lines per frame, 575 active lines per frame
- line time: 64 μ s, active line time: 52 μ s
- line scan frequency: 15.625 kHz

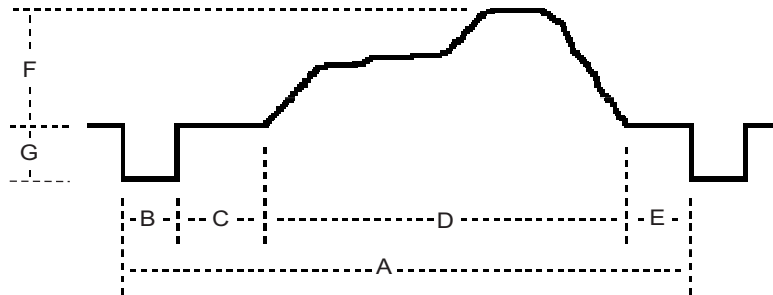
- **vertical sync**

Vertical sync refers to the vertical reference signal. It is used to vertically synchronize video devices to one another.

- **video timings**

Video timings refer to the parameters that determine how an analog or digital input signal is organized into lines, fields and frames, and where the various synchronization signals are placed. These parameters include: sync pulse width period, back porch, active period and front porch. These parameters are valid on a line basis (horizontal timings) or a field/frame basis (vertical timings).

For a typical line, these parameters refer to the following:



A. Total time for a line in a field/frame (time per line)

B. Horizontal sync pulse width

C. Horizontal back porch

D. Active line time (amount of lines that contain actual video information)

E. Horizontal front porch

The following definitions are valid for both lines (horizontal timings) or fields/frames (vertical timings):

Sync (synchronization) pulse width period: This is the state that indicates the beginning of a line (horizontal sync) or the beginning of a field (vertical sync). During this state, the video information is not valid.

Back porch: This is the state following the sync period and preceding the active video period. During this state, the video information is still not valid.

With some cameras, it might be difficult to find the duration of the sync and back porch because the camera specification describes only the timing between the beginning of the line/field and the first valid pixel/line.

Active period: This state follows the back porch period and contains the valid video information (pixels or line) generated by the camera. During this state, the digitizer grabs the video information and places it into memory.

Front porch: This is the last state in a line/field. It follows the active video period and precedes the start of the next line/field. During this state, the video information is not valid.

Note that, based on your camera type, it's possible that one or more of the above states is not present in the camera video timings. If a specific state is missing, set its corresponding timing value to zero, or the minimum allowed by Matrox Intellicam.

■ vertical video timings

Vertical video timings refer to the portion of a video signal that delimits the end of a frame/field and the beginning of a new one.

Appendix C: Video Specification Form

This appendix contains a video specification form to help you with the configuration of your digitizer.

Providing us with your video specifications

The purpose of the Video Specification Form is two-fold:

- You can use it to help identify and document the various features of your video source, making it easier for you to create your digitizer configuration file.
- You can use it to provide us with information about your video source in the event you need help creating your digitizer configuration file.

If you are having trouble getting your DCF to work properly, we would be glad to help you with it. Be sure to follow the instructions below before calling us. For phone numbers, see the **Customer Support** section at the back of this manual.

- Answer all questions in Sections I through IV directly on the form that follows. Since your camera might support several modes, your answers should reflect the desired mode of operation. Enter N/A for any non-applicable entries.
- If your camera manufacturer has provided you with detailed documentation, include a photocopy of this documentation with your completed form.
- Refer to the manual for your specific frame grabber to know if you must take any grab limitations into account.
- Fax a copy of the completed form and any other relevant documentation to the **Matrox Imaging Applications Department**, at: **(514) 969-6273**.
- ❖ The information contained in the form is the **minimum** required to determine compatibility between Matrox's imaging hardware and a specific video source.

For more details on any of the terms used in this form, refer to Appendix B.

Video Specification Form

Section I: General information

1. **Camera manufacturer:** _____
2. **Model #:** _____
3. **Camera type:**
 - ☐ Area scan ;(e.g. RS-170 camera) (aperiodic)
 - ☐ Frame scan (periodic)
 - ☐ Line scan
 - ☐ Other, specify: _____
4. **Video data rate:** _____ MHz and _____ frames/sec
5. **Desired acquisition resolution:** _____ H x _____ V
6. **Scanning format:**
 - ☐ Interlaced (2 fields per video frame)
 - ☐ Non-interlaced (1 field per video frame)
7. **Video format:**
 - ☐ Analog Amplitude _____ V (p-p)
 - Swing type:
 - ☐ positive
 - ☐ negative
 - ☐ both-swing
 - ☐ Digital _____ bits
 - ☐ TTL
 - ☐ RS-422

8. **Synchronization format:**

☐ Analog

☐ Composite video (i.e. video/sync on same wire)

☐ Video and composite sync (i.e. video & sync separate)

☐ Serrated sync

☐ Bloc sync

☐ Digital

☐ Csync

☐ Hsync and Vsync

☐ Vsync only

☐ Hsync only

Synchron. digital format:

☐ TTL

☐ RS-422

Csync or Hsync polarity:

☐ positive going polarity

☐ negative going polarity

☐ N/A

Vsync polarity:

☐ positive going polarity

☐ negative going polarity

☐ N/A

9. **Pixel clock** (if available)

clock rate: _____ MHz ☐ TTL ☐ RS-422

Section II: Description of your application

Use the space provided below to enter a detailed description of your needs (camera operation and acquisition system). Refer to the specific modes available for your camera. Draw diagrams and use extra pages as necessary.

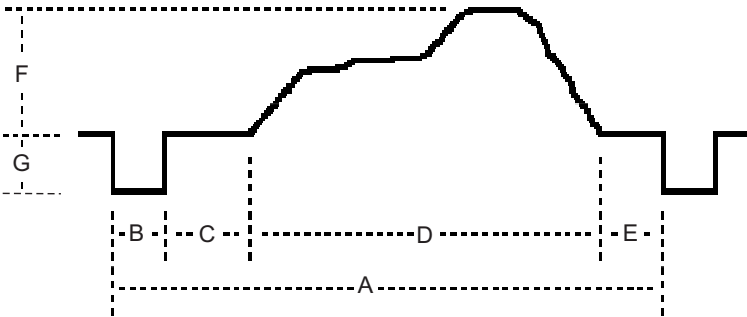
This information is important for us to choose the best way to interface your camera with our hardware.

Section III: Timings specification

Fill in the information requested below, or include the timing specification documentation provided by your camera manufacturer. **Refer to the manual for your particular frame grabber for possible grab limitations.**

Horizontal timings

Although the following diagram shows a composite video waveform, the timings also apply to separate video and sync sources.

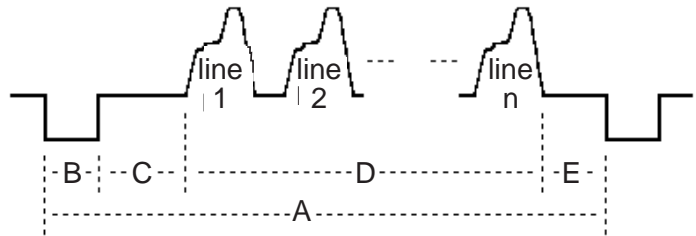


(specify whether in μsec or pixels per line)

- | | |
|---|----------------|
| A. Horiz. total line time | _____ /line |
| B. Horiz. sync pulse width | _____ /line |
| C. Horiz. back porch | _____ /line |
| D. Horiz. total active line time | _____ /line |
| E. Horiz. front porch | _____ /line |
| Horiz. values given in <input type="checkbox"/> μsec <input type="checkbox"/> pixels | |
| F. Active video amplitude | _____ V - p.p. |
| G. Sync pulse amplitude | _____ V - p.p. |

Vertical timings

For interlaced video sources, vertical timings can be expressed as the number of lines per frame or number of lines per field. For non-interlaced video sources, they should be expressed as the number of lines per frame. Note that horizontal syncs are not shown in the diagram below.



	H/field (interlaced only)	H/frame (interlaced & non-interlaced)
A. Vertical total line time	_____ lines	_____ lines
B. Vertical sync pulse width	_____ lines	_____ lines
C. Vertical back porch	_____ lines	_____ lines
D. Vertical total active time	_____ lines	_____ lines
E. Vertical front porch	_____ lines	_____ lines

Section IV: Cable specification

Use the space provided below to describe all signals accessible from your video source, including the type and gender of each physical connector on the video source (for example, female BNC connector for video; male DB-25 connector for pixel clock and synchronization). If you prefer, draw a diagram.

This information will allow us to describe the necessary cable to interface your video source with our hardware.

Estimate the minimum cable length required: _____ (feet)

Appendix D: Troubleshooting

This appendix gives suggestions to help you solve any problems you might encounter.

Troubleshooting

If you have problems using Matrox Intellicam, the following chapter might prove useful. If your problem is not addressed here or if these suggestions do not work for you, contact your local Matrox representative or the Matrox Imaging Customer Support Group.

The topics covered here are geared specifically to the **Matrox Pulsar** and the **Matrox Genesis**, unless otherwise indicated.

Keying

☛ **When I use Intellicam 2.0 with Windows 95 or Windows NT 4.0, I cannot get a stable display.**

The Microsoft PLUS program used with Windows 95 or Windows NT 4.0 remaps the colors used in each of its desktop themes. The remapped colors include all or some of the 16 reserved colors. As such, no matter what color you chose as your keying color, chances are you will run into problems with your display. In other words, you will "see through" the keying color. Experiment with various keying colors until you find one that does not interfere with the colors used in the theme you have chosen.

Grabbing

☛ **When I try to grab live with Intellicam 2.0, my grab stops or is not stable.**

If you set synchronization values that are well outside a reasonable range, the hardware will not be able to synchronize properly and will stop your grab or make it unstable.

The only way to get around this is to stop your grab completely and reallocate your DCF. Typing in your values rather than using the arrows will increase your chances of the values being accepted without problem. If you are changing all or most of your timings, you might try to stop your grab, change your timings and then re-start your grab.

Loading DCF files created with Intellicam 1.0

(**Matrox Pulsar** only)

☞ **When I try to load a DCF created with Intellicam 1.0, some of the values are converted to a new format.**

Intellicam 2.0 does convert old DCF files that contain values that are not allowed on **Matrox Pulsar**. If you do not want your original DCF to be converted, choose **Options** from the menu. Under **Preferences**, remove the check from the **Continuous Validation** checkbox.

You will still be able to change settings, but if you try to save your new configuration, you will be warned that you should not save your DCF until it has been validated.

Index

A

ABS 87
absolute value, unary expression 87
AC coupling 120
accessing Intellicam objects, MILINTER 67
accessing memory 91
ACTCM 67
active period 142, 154
active system 24
active video 142
address
 array identifier 91
 assigning 91
ADDSTR 88
ALLOCARRAY 90–91
allocate
 application 75
 buffer 75
 digitizer 75
 display 75
 memory for an array identifier 90
 system 24, 75
amplitude, video signal 118
analog video signal 116
append strings 88
application workspace, status bar 21
application, allocate 75
area, child 29, 67–68
array
 commands 90
 identifier address 91
 list, re-initializing 103
 manipulation commands 90–91
asynchronous reset trigger mode 54

B

back porch 142, 154
band (definition) 142
black reference level 149
black voltage level 143
blanking period 118–119, 143

bloc synchronization 143
breezeway, color video format 123
buffer, allocate 75
buttons, user-defined command line 29, 66

C

camera 143
 frame scan 41, 146
 interfacing 25, 41, 113
 line scan 41, 59, 136, 148
 modes of operation 128
 modes of operation, summary 139
 multi-tap 148
CCD 116
CCIR 115
CHAR, variable type 74
character array identifier, value 91
charge-coupled device 116
child area 67
 clearing, using MILINTER 68
 defining 29
choosing a system 23
 non-standard acquisition 44, 59
 standard acquisition 32
chroma bandpass 124
chroma demodulator 124
chroma trap 124
chrominance, color video format 123
clamping 120
CLEARALL 103
CLEARARRAY 91
CLEARHOST 101
clearing the MILINTER window 101
CLEARNBR 87
CLEARSTR 88
clock exchange 122, 137
CMPSTR 89
color band 142
color burst, color video format 123
color subcarrier, color video format 123
color timings 123
command block 97
command buttons, user-defined 29, 66
command history 86
command identifiers 77–78
command line, printing on the screen 104

- command list control commands 94–95
- command lists
 - characteristics 94
 - deleting from memory 94
 - editing 94, 96
 - executing 95
 - listing contents 95
 - MIL 71
 - MILINTER 78, 94
 - retrieving 95
 - saving 95
 - setting up identifiers 95
 - terminating execution 95
- command names 83
- command window, MILINTER 66
- command, delaying execution 100
- conditional operation 97
- configurable DCFs 35–36
- constants 80
- context menus 20
- context-sensitive help 35
- continuous grab 27
- Continuous Grab button 27
- conventions, notational 72
- convert
 - floating point to string 87
 - numerical expression to string 87
 - string to double 104
 - string to number 89
- creating a DCF 10, 24–25
- creating an image 26
- crystal oscillator 121
- Customer Support 156

D

- DBLTOSTR 87
- DC offset 121
- DC restoration 120
- DCF
 - asynchronous reset trigger mode 54
 - configurable 35–36
 - creating 10, 25
 - definition 24
 - digital data 49
 - document window 25
 - errors 21, 50–51

- exposure settings 57
- fine-tuning 36
- help configuring your DCF 155
- high resolution mode 47
- initial state 28
- interlaced mode 44
- modifying 10, 25
- non-interlaced mode 46
- non-standard acquisition 10, 41
- opening 25
- pre-defined files 25, 34, 42
- saving 25
- standard acquisition 25, 31
- status bar 21
- decoding color information, video format 123
- defining parameters 80
- DELAY 100
- DELCL 94
- delete
 - command lists from memory 94
 - macro identifier 93
 - macro list 93
- DELMACRO 93
- device 0 24
- digital data, frame scan 49
- digital keying 147
- digital video 115, 126
 - n-bit system 127
 - RS-422 127
 - TTL 127
- digitizer
 - allocate 75
 - controls 27
- Digitizer Control button 27
- display, allocate 75
- DO 98
- double number array, value 91
- DOUBLE, variable type 74
- dyadic operator 82
- dynamic controls 28, 38

E

- ECHO 101
- ECHOLN 101
- edit, command lists 94, 96
- EDITCL 94, 96

- electrical format
 - RS-422 145
 - TTL 145
- ELSE 97
- equalization pulses 145
- erase identifier 88
- error messages 50
 - MIL 79
 - MILINTER 106
- error types, MILINTER 79
- errors in the DCF 21, 50–51
- examples
 - asynchronous reset trigger mode 54
 - camera characteristics 43, 59
 - digital data 49
 - exposure settings 57
 - high resolution mode 47
 - MIL 28, 71
 - mstart.cl 71
 - non-interlaced mode 46
 - non-standard acquisition 41
 - Pulsar 31
 - standard acquisition 31
- EXECCL 95, 102
- exposure
 - clock 145
 - period 145
 - settings, frame scan 57
 - settings, line scan 61
 - signal 145
- expressions, numerical 81–84
- exrun.cl 71
- external parameters, MILINTER 83

F

- field 145
- file handling 102
- file path information 29
- files, supported 24, 26
- fixed line scan rate mode, line scan 136
- floating point, convert to string 87
- FOR 99
- frame 146
- frame scan camera 41, 128, 146
- freeing memory 91
- freeing objects 76

- front porch 146, 154

G

- Genesis, troubleshooting 164
- GETARRAY 91
- GETCHRARRAY 91
- GETDBARRAY 91
- GETECHOPOS 101
- GETNBR 84, 87
- GETNBRARRAY 91
- GETSHORTARRAY 91
- GETSTR 89
- GETSTRPTR 103
- GETTIMER 100
- grab
 - continuously 27, 164
 - halt 27
 - image (snapshot) 27
 - live 164
 - non-standard camera 41
 - standard camera 31
- GUI 16

H

- Halt Grab button 27
- hardware trigger 55
- Help button 18, 35
- help, command descriptions 77
- HELPPATH 103
- high resolution mode 47
- history, commands 86
- horizontal sync 146, 149
- horizontal video timings 147

I

- I/O commands 101
- IDENTIFIER 95
- identifier 85
 - add string value 89
 - assigning a definition 93
 - assigning an address 91
 - erase 87–88
 - freeing associated memory 91
 - macros 88

- syntax 85
- IF 97
- image
 - accessing with MILINTER 69
 - attributes 26
 - creating 26
 - filling with current digitizer sizes 26
 - grab (snapshot) 27
 - status bar 21
- information messages, status bar 21
- initial DCF state 28
- INKEY 101
- insert mode 86
- INSTR 101
- Intellicam
 - allocated application object, MIL IDs 70
 - allocated system objects, MIL IDs 70
 - application object, macros 70
 - dialog boxes 22
 - images, MIL IDs 69
 - installing 12
 - loading 17
 - objects, accessing 67
 - objects, freeing 67, 76
 - objects, MIL IDs 67
 - on-line help 22
 - overview 10
 - requirements 11
 - supported files 24, 26
 - system objects, accessing 70
 - system objects, macros 70
 - tips 17
- interfacing a camera 25, 41, 113
- interlaced mode 116, 147
- internal commands, MILINTER 78

K

- KEYHIT 101
- keying 147, 164

L

- LEFTSTR 89
- LENSTR 89
- line editor, MILINTER 85
- line scan camera 41, 59, 148

- LISTCL 95
- LOADCL 95, 102
- LOADMACRO 93, 102
- long number array, value 91
- LONG, variable type 74
- loops 98
- luminance, color video format 123

M

- MACRO 93
- macro commands 93
- macro identifier 88
 - deleting 93
 - redefining 93
- macro list
 - deleting 93
 - retrieving 93
 - saving 93
- macros 67, 69, 75–76
 - allocated Intellicam application object 70
 - allocated Intellicam system objects 70
 - opened image objects 69
- MBufClear() 68
- MDI 16
- memory
 - accessing 91
 - freeing 91
- menu bar 20
- messages
 - information 21
 - status 21
- MIDSTR 89
- MIL

- accessing functions 64, 66–67
- command lists 71
- environment, setting up 75
- error messages 79
- examples 28, 71
- functions, accessing 28
- library functions 78
- objects, freeing 67
- standard types and macros 73
- MIL IDs
 - allocated application objects 70
 - images 69
 - Intellicam objects 67

- system objects 70
- variable type 73
- MILINTER 28
 - accessing Intellicam objects 67
 - buttons 65
 - camera, defining 72
 - child area, clearing 68
 - command buttons 66
 - command identifiers 77
 - command lists 65, 78, 94
 - command window 66–67, 69
 - command window, clearing 101
 - command window, closing 103
 - commands 66
 - default system, defining 72
 - error messages 79, 84, 106
 - error types 79
 - external parameters 83
 - file path information 29
 - help files (directory) 66, 103
 - image files (directory) 66
 - image, defining 72
 - internal commands 78
 - internal timer, resetting 100
 - invoking 65
 - loops 98
 - macros 67, 69, 75–76, 78, 93
 - macros, allocated application objects 70
 - macros, allocated Intellicam system objects 70
 - macros, opened image objects 69
 - MIL Interpreter button 29
 - milinter.ini 65, 102
 - overview 65
 - printing a string 101
 - start-up file 29, 147
 - start-up file (filename & directory) 65
 - syntax 28, 64, 111
 - user-defined command buttons 29
- milinter.ini 65
- MIL-Lite, accessing functions 64, 66–67
- milsetup.cl 72, 75
- miscellaneous commands 103
- modes of operation, camera 128–132, 135–137
- modes of operation, camera (summary) 139
- monadic operator 81
- mstart.cl 71

- Multiple Document Interface 16
- multi-tap camera 148

N

- NBRTOSTR 87
- negative-going video 115, 125
- New DCF button 25
- New Image button 26
- non-interlaced mode 46, 116, 148
- non-standard acquisition 10, 41
- non-standard timings 115
- notational conventions 72
- Notepad, Windowseditor' 96
- NTSC 115
- number identifier 87
- number list, re-initializing 103
- numerical expression 84
 - assigning to a character array identifier 92
 - assigning to number array identifier 92
 - convert to string 87

O

- objects
 - Intellicam 67
 - sharing 75
- ONBREAK 95
- on-line help 22, 35
- Open button 25
- OS shell 105
- overflows 83
- overwrite mode 86

P

- PAL 115
- parameter
 - definition 80
 - out of range 83
 - syntax 80
- PAUSE 101
- pedestal 148
- phase-locked loop 121
- photosensors 116
- pixel clock 121, 148
- PLL 121

- pointer, to a string 103
- polarity 149
- pre-defined DCF files 25, 34, 42
- printing a string 101
- printing position, changing 101
- processing environment, setting up 75
- progressive scan mode 116
- pseudo-continuous mode, frame scan camera 130
- Pulsar
 - examples 33
 - RS-422 digital module 49
 - troubleshooting 164

Q

QUIT 103

R

RAND 88

- random value 88
- reference levels 149
- re-initializing lists 103
- RESERVED 78, 103
- reserved words, listing 103
- RETURN 95
- return values, printing on the screen 104
- RIGHTSTR 89
- RS-170 115
- RS-330 115
- RS-343 115, 143
- RS-422 50, 127, 145
- running MIL command lists 71

S

- sample program 71
- sampling rate 121, 149
- Save button 25
- SAVECL 95, 102
- saveCL 71
- SAVEMACRO 93, 102
- SECAM 115
- serration pulses 149
- SETARRAY 91
- SETCHRARRAY 92

- SETDBARRAY 92
- SETECHOPOS 101
- SETNBR 88
- SETNBRARRAY 92
- SETSHORTARRAY 92
- SETSTR 84, 88–89
- SETTIMER 100
- sharing objects 75
- short number array, value 91
- SHORT, variable type 74
- SILENCE 104
- Single Grab button 18, 27
- start-up file, MILINTER 29, 65, 147
- status bars 21
- status messages 21
- string
 - append 88
 - compare strings 89
 - convert to double 104
 - convert to number 89
 - determine middle m characters 89
 - determine n leftmost characters 89
 - determine number of characters 89
 - identifier, string value 89
 - list, re-initializing 103
 - manipulation commands 88
 - parameter, as a command 88
 - value, assigning to an identifier 101
- strings 84
- STRTODBL 104
- STRTONBR 89
- swing type 150
- sync
 - bloc 143
 - clip level 117
 - horizontal 146, 149
 - pulse period 149, 154
 - signal 117, 149
 - vertical 149, 153
- syntax
 - MILINTER 28, 64, 111
 - parameter 80
- system
 - active 24
 - allocate 24, 75
 - choosing 23, 32
 - default 24
 - definition 23, 150

- device 0 24
- non-standard acquisition 44
- SYSTEM command 105
- virtual system 23–24

System Selection button 23

T

terminating execution, command list 95

threads 67

timer 100, 151

Tip of the Day 17

tool button bar 18–19

TRACE 104

trigger mode, frame scan 131

trigger, hardware 55

TTL 127, 145

U

unary expression, absolute value 87

user-defined command line buttons 29, 66

V

variable line scan rate mode, line scan 137

variable manipulation 87

VERBOSE 104

vertical sync 149, 153

vertical video timings 154

video format 151

- color 123

- non-standard 115, 125

- standard 115

video signal

- amplitude 118

- analog 116

- composite 117

- composite color signal 124

Video Specification Form 42, 155

video timings 25, 153

virtual system 23–24

W

WHILE 98

white reference level 149

window management 22

Windows conventions 16

Product Support

Product Assistance Request Form

[illegible]

