SNAPPER Developer's Guide

DataCell Limited

v4.0.1

Disclaimer

While every precaution has been taken in the preparation of this manual, DataCell Ltd assumes no responsibility for errors or omissions. DataCell Ltd reserves the right to change the specification of the product described within this manual and the manual itself at any time without notice and without obligation of DataCell Ltd to notify any person of such revisions or changes.

Copyright Notice

Trademarks

"Apple", "Macintosh" and "MacOS" are trademarks of Apple Computer Inc. "AMCC" is a registered trademark of Applied Micro Circuits Corporation. "Dallas" is a registered trademark of Dallas Semiconductor Corporation. "Dell" is a registered trademark of Dell Computer Corporation. "IBM", "PC/AT", "PowerPC" and "VGA" are registered trademarks of International Business Machine Corporation. "MetroWerks" and "CodeWarrior" are registered trademarks of MetroWerks Inc. "Microsoft", "CodeView", "MS" and "MS-DOS", "Windows", "Windows NT", "Windows 95", "Windows 98", "Win32", "Visual C++" are trademarks or registered trademarks of Microsoft Corporation. "National Semiconductor" is a registered trademark of National Semiconductor Corporation. "Sun", "Ultra AX" and "Solaris" are registered trademarks of Sun Microsystems Inc. All "SPARC" trademarks are trademarks or registered trademarks of SPARC International Inc. "VxWorks" and "Tornado" are registered trademarks of Wind River Systems Inc. "Xilinx" is a registered trademark of Xilinx.
All other trademarks and registered trademarks are the property of their respective owners.

Part Information

Part Number: SNP-MAN-DEVGUIDE

Version v4.0.1 September 1999

Printed in the United Kingdom.

Contact Details

Europe & ROW	Web	www.datacell.co.uk
	Sales	info@datacell.co.uk
	Support	techsupport@datacell.co.uk
USA	Web	www.datacell.com
	Sales	info@datacell.com
	Support	techsupport@datacell.com

Head Office:
DataCell Limited.
Falcon Business Park, 40 Ivanhoe Road,
Finchampstead, Berkshire, RG40 4QQ, UK

Tel +44 (0) 1189 324324
Fax +44 (0) 1189 324325

Table of Contents

Operating System Specific Information

This section gives an overview of the Snapper SDK under each supported operating system and some useful programming notes. Comprehensive example applications with full source code are provided with the SDK, which are probably the best starting point for the development of custom applications. All platforms, and operating systems are covered here, therefore you probably only need to refer to the section that is relevant to you. Please also refer to the "TMG Library Programmer's Manual", which covers many of the concepts applicable throughout the Snapper libraries.

There are sections devoted to each operating system, that is MS-DOS, Windows 3.1 (including Windows 3.11), Windows 95 (including Windows 98), Windows NT, Solaris 2, MacOS, VxWorks and LynxOS. There is an additional section on "datamappers" towards the end of this section, which outlines a programming methodology to make full use of the datamapper's flexibility.

The Installation Guide lists the hardware system requirements for each operating system.

The Snapper SDK datasheet (in the Appendices) lists the supported compilers.

MS-DOS PROGRAMMING

SDK Structure After Installation

By default, the software is installed into the directory *c:\snapdos*. Below this directory there are two directories:

- *apps* - This contains applications with full source code.
- *lib* - This directory contains two sub-directories – dos and src. dos contains the MS-DOS static libraries for real and protected mode applications; src contains a sub-directory for the include files and a sub-directory for some relevant source code.

Programming Notes

32 bit protected mode static libraries are provided for MS-DOS, using the Symantec DOSX (32 bit flat) memory model and the Watcom 32 bit flat model.

The protected mode libraries consist of the following two libraries:

- *tmgx.lib* - TMG library;
- *snapx.lib* - Snapper and Bus Interface Board library,

where the "x" is replaced with "p" for Symantec and "w" for Watcom. When using Symantec, the supported memory model is DOSX ("-mx" option). Under Watcom, the supported memory model is the 32 bit flat model ("-mf" option). To provide compatibility with the DOS display functions (based on the Flash Graphics library), Watcom applications must be linked with the X-32VM DOS extender (Snapper part number X32-DOS-LIB).

The TMG library provides a simple, yet powerful API for the display of images based on the Flash Graphics library (by Flashtek and available as a separate item, Snapper part number FG-DOS-LIB). See the TMG Library Programmer's Manual for more details.

Several comprehensive application examples are provided with the SDK under the *apps* directory that show how to use the libraries and each Snapper. The applications are:

- *process* - This application is the simplest example which uses no hardware, but shows how to use the TMG library by reading in an image and then writing it out laterally inverted.
- *pgen* - This is a simple example which shows how to generate an image test pattern and save it as a TIFF file. It also illustrates how individual pixels may be accessed.
- *view16* - A image file viewer that displays the image in 16 bits per colour.
- *jview* - A JPEG image file viewer that can use either hardware or software JPEG decompression.
- *comp* and *compsw* - JPEG compression examples using hardware and software respectively.

- *decomp* and *decompsw* - JPEG decompression examples using hardware and software respectively.
- *s16dos* - Snapper-16 demonstration application.
- *s24dos* - Snapper-24 and Snapper-8 demonstration application.
- *d16dos* - Snapper-DIG16 demonstration application.
- *gamma*, *chroma*, *seq* - Three advanced applications showing the use of: the TMG_LUT functions; the TMG chroma keying functions; and the Snapper/TMG motion JPEG functions.

The simple examples, such as *process* and *pgen,* only need to be compiled and linked with the TMG library. The compiler pre-processor directive *_DOS32* needs to be defined.

view16 requires the Flash Graphics library and must also be compiled with *fg_cstm.c* and *tmg_dos.c*. These additional files are not compiled into the TMG library so that applications can be built without needing the Flash Graphics library. An additional compiler pre-processor directive, *_FG_GRAPHICS,* is now needed.

The example applications s*16dos, s24dos* and *d16dos*, and associated makefiles, provide a good example for doing many Snapper operations. They each have a simple instruction guide on how to drive them that can be found in the end of the Installation section. They are probably the best starting point for anyone developing custom MS-DOS applications.

The compiler pre-processor directives are summarised below:

_DOS32	For 32 bit protected mode compilations - using the Symantec or Watcom compilers.
_FG_GRAPHICS (or just _FG)	Use the Flash Graphics library. This is a pre-requisite for MS-DOS imaging applications. This is a low cost, yet comprehensive, royalty free graphics library (Snapper part number FG-DOS-LIB). It is used to display images to VESA compatible graphics cards.

Acquisition from PCI Snappers use DMA under protected mode DOS, although it reverts to programmed I/O if the application is running under a DPMI server - for example in a DOS box under Windows 3.1.

WINDOWS 3.1X PROGRAMMING

SDK Structure After Installation

By default, the software is installed into the directory *c:\snapw31*. Below this directory there are two sub-directories:

- *apps* - This contains applications with full source code and on-line help.
- *lib* - This contains two sub-directories - dll and src. dll contains a copy of the DLLs that have been installed in the Windows system directory; src contains a sub-directory for the include files and a sub-directory for some limited source code.

Programming Notes

The Snapper libraries are provided as two DLLs - firstly, the Snapper DLL, *snapw31.dll,* and secondly, the TMG library DLL, *tmgw31.dll*. These DLLs are copied into your Windows system directory automatically during installation. To provide DCI (Display Control Interface) support, the standard DCI manager DLL, *dciman.dll*, is also installed.

The TMG library provides a simple, yet powerful API for the display of images (which is consistent across all supported operating systems). See the "TMG Library Programmer's Manual" for more details.

Four comprehensive application examples are provided with the SDK under the *apps* directory that show how to use the libraries and each Snapper. These applications have all been built using Microsoft Visual C++ based on the MFC application framework. The applications are:

- *IMV* - This application is an image viewer that will load and display all supported image file formats. It can also perform JPEG compression and decompression of images using either the TMG software library or the Crunch JPEG Bus Interface Board.
- *S16* – Snapper-16 demonstration application.
- *S24* – Snapper-24 and Snapper-8 demonstration application.
- *D16* – Snapper-DIG16 demonstration application.

All the applications have on-line help that describes each control in detail.

In additional to these applications, all the MS-DOS applications are also provided - see the "MS-DOS Programming" section for details on these applications. Although they are not designed for Windows 3.1, they contain useful example code illustrating how to use certain areas within the Snapper libraries.

Applications should be compiled using the large memory model with the pre-processor compiler directive *_WIN31* defined.

For users new to Windows 3.1 programming (as well as for more experienced programmers) the following book is recommended: "Inside Visual C++" by David J Kruglinski, second edition (Version 1.5) by Microsoft Press. ISBN 1-55615-661-8.

WINDOWS 95 PROGRAMMING

SDK Structure After Installation

By default, the software is installed into the directory *c:\Program Files\Snapper\Win95\x86\SDKx.y.z,* where x.y.z are the major, minor and sub-minor revision levels respectively. Below this directory there are three sub-directories:

- *apps* - This contains applications with full source code and on-line help.
- *help* - This contains on-line help for all the libraries within individual sub directories for each file format, ie Win32 help, HTML & PDF.
- *lib* - This contains two sub-directories – system and src. system contains a copy of the DLLs and VxD files that have been installed in the Windows system directory; src contains a sub-directory for the include files and a sub-directory for some limited source code.

Programming Notes

The Snapper libraries are provided as three DLLs:

- *snap95.dll* - This DLL contains all the Snapper-8/24, Snapper-16, Snapper-Dig16 and Bus Interface Board library code.
- *fpga95.dll* - This DLL contains all the configuration files for all the FPGAs (Field Programmable Gate Arrays) which are used to implement the Datamappings (see page 11) as well as hardware control functions.
- *tmg95.dll* - This DLL contains all the TMG imaging library functions, which provides a simple, yet powerful API for the display of images (which is consistent across all supported operating systems). See the "TMG Library Programmer's Manual" for more details.

Four comprehensive application examples are provided with the SDK under the *apps* directory that show how to use the libraries and each Snapper. These applications have all been built using Microsoft Visual C++ based on the MFC application framework. The applications are:

- *IMV* - This application is an image viewer that will load and display all supported image file formats. It can also perform JPEG compression and decompression of images using either the TMG software library or the Crunch JPEG Bus Interface Board.
- *S16* – Snapper-16 demonstration application.
- *S24* – Snapper-24 and Snapper-8 demonstration application.

- *D16* – Snapper-DIG16 demonstration application.

All the applications have on-line help that describes each control in detail. In addition, some simple cross platform example applications are provided, see page 13.

Applications should be compiled using the large memory model with the pre-processor compiler directive *_WIN95* defined.

WINDOWS NT PROGRAMMING

SDK Structure After Installation

By default, the software is installed into the directory *c:\Program Files\Snapper\WinNT\x86\SDKx.y.z,* where x.y.z are the major, minor and sub-minor revision levels respectively. Below this directory there are three sub-directories:

- *apps* - This contains applications with full source code and on-line help.
- *help* - This contains on-line help for all the libraries within individual sub directories for each file format, ie Win32 help, HTML & PDF.
- *lib* - This contains two sub-directories – system32 and src. system32 contains a copy of the DLLs and SYS files that have been installed in the Windows system directory and drivers directories respectively; src contains a sub-directory for the include files and a sub-directory for some limited source code.

Programming Notes

The Snapper libraries are provided as three DLLs:

- *snapNT.dll* - This DLL contains all the Snapper-8/24, Snapper-16, Snapper-Dig16 and Bus Interface Board library code.
- *fpgaNT.dll* - This DLL contains all the configuration files for all the FPGAs (Field Programmable Gate Arrays) which are used to implement the Datamappings (see page 11) as well as hardware control functions.
- *tmgNT.dll* - This DLL contains all the TMG imaging library functions, which provides a simple, yet powerful API for the display of images (which is consistent across all supported operating systems). See the "TMG Library Programmer's Manual" for more details.

Four comprehensive application examples are provided with the SDK under the *apps* directory that show how to use the libraries and each Snapper. These applications have all been built using Microsoft Visual C++ based on the MFC application framework. The applications are:

- *IMV* - This application is an image viewer that will load and display all supported image file formats. It can also perform JPEG compression and decompression of images using either the TMG software library or the Crunch JPEG Bus Interface Board.
- *S16* – Snapper-16 demonstration application.
- *S24* – Snapper-24 and Snapper-8 demonstration application.
- *D16* – Snapper-DIG16 demonstration application.

All the applications have on-line help that describes each control in detail. In addition, some simple cross platform example applications are provided, see page 13.

Applications should be compiled using the large memory model with the pre-processor compiler directive *_WINNT* defined.

SOLARIS 2 PROGRAMMING (SPARC)

SDK Structure After Installation

After installation of the full SDK the directory */opt/ASLsnap* will have the following main sub-directories:

- *apps/src* - This directory contains the application examples and utilities with full source code.
- *apps/bin/solaris/sparc* – This directory contains the pre-compiled application binaries.
- *lib/solaris/sparc* - This directory contains the shareable object ".so" libraries.
- *lib/src* - This directory contains a sub-directory for the include files and sub-directories for some relevant source code.
- *help* - This contains on-line help for all the libraries within individual sub directories for each file format, ie Win32 help, HTML & PDF

It should be noted that all the pre-compiled binaries and supplied makefiles assume that the shared object libraries are in the default location of */opt/ASLsnap/lib/solaris/sparc*. If this is not the case, the actual location of the shareable object modules must be specified on the *LD_LIBRARY_PATH*

Programming Notes

The following shareable libraries are supplied.

- *libtmg.so* - TMG library.
- *libsnap.so* - Snapper and Bus Interface Board library.
- *libfpga*.so – Hardware configuration files.

The TMG library provides a simple, yet powerful API for the display of images (which is consistent across all supported operating systems). See the "TMG Library Programmer's Manual" for more details.

Several comprehensive application examples are provided with the SDK under the *apps* directory that show how to use the libraries and each Snapper. The applications are:

- *s16sol* - A very simple Snapper-16 demonstration application.
- *s24sol* - A very simple Snapper-24 and Snapper-8 demonstration application.
- *d16sol* - A very simple Snapper-DIG16 demonstration application.

The compiler pre-processor directives are summarised below:

_SOLARIS2	For Solaris 2 compilations.
_X_GRAPHICS	This is required if displaying under the X windows environment.

The makefiles with the application source are supplied to work with SunSoft 'C' V4.2 or later, but only 2 lines need changing to use GNU C - see the comments in the makefile for what to change.

Using Multiple Threads

The libraries are linked in a manner that allows the use of multi-threaded applications. However, when writing applications to take advantage of multiple threads two restrictions need to be considered.

1. The Snapper hardware is inherently a non-sharable resource - that is, it is physically impossible for two threads to both be doing independent captures at the same time. If an attempt is made to do this it is treated as an error, and no attempt is made at the library or device driver level to queue the requests.

2. All functions in the TMG and Snapper libraries which use strip processing (i.e. have a *TMG_action* parameter) are not re-entrant when processing in strips. To use these functions in a multi-threaded environment the strip size must be set to the total image size.

It is recommended that when using the Snapper libraries in a multi-threaded application, a mutex lock is created which prevents multiple threads from executing concurrently in the Snapper libraries. It is permissible to have separate threads operating concurrently on different base boards, but care should still be taken that only one thread calls any TMG function at any time.

Note that both the Dig16 and Snap24 libraries use multiple threads internally. This is particularly important to bear in mind when using certain debugging tools which lock the thread being debugged, thus not allowing any background threads to execute.

VXWORKS PROGRAMMING (POWERPC)

SDK Structure After Installation

1. Solaris Hosts

After installation of the full SDK the directory */opt/ASLsnap* will have the following main sub-directories:

- *apps/src* - This directory contains the application examples and utilities with full source code.
- *apps/bin/vxworks/ppc603* – This directory contains the pre-compiled PowerPC603 application binaries.
- *apps/bin/vxworks/ppc604* – This directory contains the pre-compiled PowerPC604 application binaries.
- *lib/vxworks/ppc603* - This directory contains the static libraries for PowerPC603 processor series
- *lib/vxworks/ppc604* - This directory contains the static libraries for PowerPC604 processor series
- *lib/src* - This directory contains a sub-directory for the include files and sub-directories for some relevant source code.
- *help* - This contains on-line help for all the libraries within individual sub directories for each file format, ie Win32 help, HTML & PDF

2. NT Hosts

After installation of the full SDK the directory *c:\SnapperVxWorks* will have the following main sub-directories:

- *apps\src* - This directory contains the application examples and utilities with full source code.
- *apps\bin\ppc603* – This directory contains the pre-compiled PowerPC603 application binaries.
- *apps\bin\ppc604* – This directory contains the pre-compiled PowerPC604 application binaries.
- *lib\ppc603* - This directory contains the static libraries for PowerPC603 processor series
- *lib\ppc604* - This directory contains the static libraries for PowerPC604 processor series
- *lib\src* - This directory contains a sub-directory for the include files and sub-directories for some relevant source code.
- *help* - This contains on-line help for all the libraries within individual sub directories for each file format, ie Win32 help, HTML & PDF

Programming Notes

The following shareable libraries are supplied.

- libtmg.a - TMG library.
- libsnap.a - Snapper and Bus Interface Board library.
- libfpga.a – Hardware configuration files.
- libsnapdrv.a – Low level VxWorks specific routines. This replaces the separate driver layer of most other operating systems.

The TMG library provides a simple, yet powerful API for the display of images (which is consistent across all supported operating systems). See the "TMG Library Programmer's Manual" for more details.

The compiler pre-processor directives are summarised below:

_VXWORKS	For all Snapper VxWorks compilations.
CPU=603	For all PowerPC 603 target processors
CPU=604	For all PowerPC 604 target processors

The actual driver installation process is highly Board Support Package dependent. Several example *InstallSnapper()* routines have been provided in the sample application programs. Refer to the VxWorks programmers' manual for further information.

Using Multiple Threads

The libraries are linked in a manner that allows the use of multi-threaded applications. However, when writing applications to take advantage of multiple threads two restrictions need to be considered.

1. The Snapper hardware is inherently a non-sharable resource - that is it is physically impossible for two threads to both be doing independent captures at the same time. If an attempt is made to do this it is treated as a error, and no attempt is made at the library or device driver level to queue the requests.
2. All functions in the TMG and Snapper libraries which use strip processing (i.e. have a *TMG_action* parameter) are not re-entrant when processing in strips. To use these functions in a multi-threaded environment the strip size must be set to the total image size.

It is recommended that when using the Snapper libraries in a multi-threaded application, a mutex lock is created which prevents multiple threads from executing concurrently in the Snapper libraries. It is permissible to have separate threads operating concurrently on different base boards, but care should still be taken that only one thread calls any TMG function at any time.

Note that both the Dig16 and Snap24 libraries use multiple threads internally. This is particularly important to bear in mind when using certain debugging tools which lock the thread being debugged, thus not allowing any background threads to execute. The internal threads are required to pre-empt the thread normally running the Snapper call. In order to do this, the internal threads are created at a priority equal to the priority of the thread calling *BASE_create* minus one. Thus it is important that the priority of the calling thread is not boosted (priority value decreased) between the call to *BASE_create* and any subsequent Snapper calls.

LYNXOS PROGRAMMING

SDK Structure After Installation

The software is installed into the directory selected by the user. Below this directory there are three sub-directories:

- *apps* - This contains applications with full source code and on-line help.
- *help* - This contains on-line help for the SDK libraries in HTML format.
- *lib* - This contains two sub-directories – lib and src. lib contains the object files in a further sub-directory lib/rel. Also in this directory are the driver object files and instructions for its use as either statically linked or dynamically linked to the kernel.
- *src* contains a sub-directory for the include files and a sub-directory for some limited source code.

Programming Notes

The Snapper libraries are provided as three '.lib' files:

- *libsnap.a* - This library contains all the Snapper-8/24, Snapper-16, Snapper-Dig16 and Bus Interface Board library code.
- *libfpga.a* - This library contains all the configuration files for all the FPGAs (Field Programmable Gate Arrays) which are used to implement the Datamappings (see page 11) as well as hardware control functions.
- *libtmg.a*. - This library contains all the TMG imaging library functions, which provides a simple, yet powerful API for the display of images (which is consistent across all supported operating systems). See the "TMG Library Programmer's Manual" for more details.

The driver is provided in two versions with instructions for installation found in the readme.txt file in the /lib/lib/rel directory:

- *Snapdrv.o* - the object file for installation as a statically linked kernel driver. The support file snapper.cfg is also supplied.
- *Snapdrv.obj* - the object file for a dynamically installed kernel driver.

Comprehensive application examples are provided with the SDK under the *apps* directory that show how to use the libraries and each Snapper. These applications have all been built and tested on LynxOS machines

The compiler pre-processor directives are summarised below:

_X_GRAPHICS	This is required if displaying in an X-Windows environment. The libraries are compiled by default with X support built in. To negate this, compile the file xlibstub.c to create libX.a and link with that.

The makefiles with the application source work with the OS Cygnus compiler and GNU C compilers.

MACOS PROGRAMMING

SDK Structure After Installation

After installation of the full SDK the root folder will have five main sub-folders:

- *DropOntoSystemFolder* - This folder contains the driver and shared library files.
- *StaticLibs* – Contained within this folder are all the shared libraries mentioned above, in a static library form.
- *Headers* –This folder contains all the necessary header files.
- *Source* – Library source code is contained within this folder.
- *Applications* - This directory contains the application examples and utilities with full source code.

Programming Notes

The driver files contained within DropOntoSystemFolder must be copied to the System folder. The Mac must then be rebooted to allow the driver to install. (Mac drivers are only installed at boot time – without re-booting the applications will run with the existing driver, if one exists, and not function correctly). It is recommended that the shared library files are also copied to the System folder, which then allows then to be globally accessed by any application on the Mac. However, if preferred, they can be placed in the same directory as the application.

The static libraries are generated from the same library source code as the shared libraries. They are intended for users who wish to generate a single executable which contains the application and library information. This eliminates the risk that a user can have incompatible application and library code, but does increase the size of each application.

The Headers folder contains two sub folders; SnapperHeaders and MacOsHeaders. SnapperHeaders contains all the include files specific to the Snapper SDK, whereas MacOsHeaders contains files supplied by Apple for additional file management support. Both folders must be included in a project, when compiling and linking with the Snapper SDK.

The *Source* folder contains further sub folders for Snapper-8/24 and Snapper-Dig16. Within these sub folders are supplied the source code to the library initialisation functions. These are provided as a basis for users who wish to write custom initialisation code, for non standard video sources.

The *Applications* folder contains full source code to simple example programs for each of the Snapper products. These are intended to show the basic principles of operation, and as such do not exercise all the available features.

Cross Platform Information

This section contains information which applies to all (or many) operating systems.

In order to use common library source code, the Snapper libraries use platform independent data types which are described in the *Integer Types* section. *DataMappers* describes the proprietary hardware which allows pixel formats to be remapped, thus reducing the processing time required to process or display an image.

Example Applications describes a suite of cross platform applications which provide simple examples of how to acquire images from the Snapper hardware. They can also be used to compare the relative benefits of the different modes of operation.

INTEGER TYPES

Sizes of integers vary between compilers and operating systems and are a potential source of portability errors. To overcome this the following types are used and are constant across all compilers and operating systems:

ui8	8 bit unsigned integer (unsigned char)
i8	8 bit signed integer (char)
ui16	16 bit unsigned integer
i16	16 bit signed integer etc.
ui32	32 bit unsigned integer
i32	32 bit unsigned integer

For pointers to image data the following types are used:

*IM_UI8**	Pointer to an 8 bit unsigned integer
*IM_UI16**	Pointer to a 16 bit unsigned integer
*IM_UI32**	Pointer to a 32 bit unsigned integer.

These are actually the same as the basic data types above (i.e. *IM_UI8* = ui8**) under all operating systems apart from Windows 3.1x. Under Window 3.1x these types include the *_huge* modifier that allows the pointer to auto-increment across a 64K memory boundary. Note that the *_huge* modifier only modifies the variable to its immediate right, so the following code will fail:

```
/* Only pData1 is modified to __huge */
IM_UI8 *pData1, *pData2;
```

The correct definition is as follows:

```
IM_UI8 *Pdata1;
IM_UI8 *Pdata2;
```

Under Windows 3.1x and real mode MS-DOS the large memory model should be used.

Another potential pitfall is the following code that is fine under 32 bit compilers, but will fail under real mode MS-DOS:

```
int num_of_pixels;
num_of_pixels = image_width * image_height;
```

But under 16 bit (real mode) MS-DOS, the variable `num_of_pixels` is only 16 bits and would probably not be large enough. In this example `num_of_pixels` should be a *"ui32"*.

All these general type definitions are in the file "os_gen.h" which is supplied with the Snapper SDK.

See also the "TMG Library Programmer's Manual".

DATAMAPPERS

A Snapper datamapper is a proprietary reprogrammable hardware component that provides enormous flexibility over the type of acquisition pixel format. It is fitted on the PCI and SBus (but not ISA) Bus Interface Boards in the datapath so that the pixel format can be remapped "on the fly" whilst acquiring image data. The real benefit comes from the fact that this datamapper is reprogrammable via software. The Snapper library functions, *SNPxx_set_format,* configure this device to map the image data to the selected pixel format.

The configuration data for the datamappers is generally stored in a separate library file, or occasionally in the Snapper library file. As there are a large number of pixel mappings supported, the size of this stored information can be large. If the application space is limited, ie in embedded systems such as VxWorks or LynxOS, then only those required mappings should be linked from the static library. Similarly, if the embedded application will only be running on a particular Snapper, then control files for other Snappers can be omitted from the link. See the *readme.txt* file in the FPGA directory for a description of the purpose of each file.

Pixel formats are described in detail in the "TMG Library Programmer's Manual", but essentially they are the different ways in which a pixel can be represented - for example colour pixels are usually represented by 32, 24, 16 or 15 bits of binary data. The purpose of the datamapper is to map the raw image data into a format suitable for the application. For example to achieve fast display update rates it would be desirable to read the pixels in the same format as the display hardware. This could be 15 bit colour (Windows 32k colour mode) or 16 bit colour etc. So, for example, a 15 bit datamapper would be installed. When an image needs to be saved, the datamapper should be set back to 8 bits per colour plane so that the image is stored in high quality (e.g. 24 bits per pixel for colour). Note that this datamapper philosophy does not compromise the use of various Windows display modes in order to achieve high speed image update.

The pixel mappings could of course all be done in software - for example colour data could always be acquired at 32 bits per pixel, then the function *TMG_image_convert* used to convert the format to that of the display (or whatever). In fact *TMG_image_convert* could be thought of as a software datamapper (although it is slightly more powerful than the its hardware counterpart).

The following "pseudo" code fragment suggests a methodology for using the datamappers to achieve high speed display update that has the following benefits; it works on ISA and PCI Bus Interface Boards; it works if there are no datamappers present (but slower); and most importantly, it works faster if a datamapper is subsequently added (without any code change). This is the principle that is used in the Windows 3.1x/9x/NT applications examples. The example below assumes colour acquisition using Snapper-24 on either the ISA or PCI Bus Interface Boards. Please refer to the actual code in the Microsoft Visual C++ project *s24*.

This first code fragment function would be used after initialisation of the Snapper as part of the set up procedure.

```
/* switch off library error returns in case set_format fails -
 * which it may do if there is not a datamapper file for the
 * requested mode.
 */
Switch off error reporting
AcquisitionFormat = Read display pixel format

/* match acquisition pixel format to display pixel format */
Use SNPxx_set_format to set the acquisition format to AcquisitionFormat.

If the set_format command failed {
   print "We don't have a datamapper - update will be slower"

   /* RGBX32 always works for both ISA and PCI */
   AcquisitionFormat = TMG_RGBX32;
   result = SNP24_set_format(S24.m_hSnapper, SNP24_FORMAT_RGB,
       S24.m_AcquisitionFormat);
}
Set error reporting on using default handler
```

This next code fragment would be used in the capture/display loop:

```
Read the video data
Start the next capture
if (AcquisitionFormat != DisplayPixelFormat)
   TMG_image_convert to display format
else
   Simply display it
```

Note that if it was necessary to save an image when running live in colour using 16 bits per pixel, it would be necessary to temporarily load a 32 bit datamapper so as to acquire a full 24 bit colour image and then switch back to 16 bit, to continue with fast display update.

VISUAL BASIC FOR WINDOWS 3.1X

By making use of the standard Snapper Windows DLLs, it is possible to write Visual Basic applications to control Snapper that run under Windows 3.1x. All that is required is to provide definitions for the required Snapper function calls, and to specify in which DLL they are implemented. (All functions that begin "TMG_" are included in *tmgxx.dll*, and all the others are in *snapxx.dll*). As a result, the Visual Basic programmer is expected to have a limited knowledge of the C language in order to understand the syntax of the supplied Snapper header files.

For example, in order to call *SNP24_capture*, the following definition must be included as one line in the Visual Basic Declarations:

```
Declare Function SNP24_capture Lib "snapw31.dll" (ByVal hSnp24 As Long, ByVal
      Mode As Long) As Long
```

This refers to a function called *SNP24_capture* in the Snapper DLL, which accepts two parameters of type *Long*, and returns a parameter of type *Long*. This information is obtained by referring to the definition of *SNP24_capture* in the "Snapper-24 Programmer's Manual":

```
Terr EXPORT_FN SNP24_capture(Thandle, Tparam);
```

and using the following type conversions between C and Visual Basic:

i32, ui32, Tboolean, Terr, Thandle, Tparam	⇔	Long
ui16, i16	⇔	Integer

The *Mode* value passed to *SNP24_capture* is a software token. It is also recommended (but not obligatory) that the software tokens are included in the Visual Basic Declarations section. For example, the following C code in *\include\snp24.h*

```
#define SNP24_START_AND_WAIT        ((Tparam) 0x0000)
#define SNP24_START_AND_RETURN      ((Tparam) 0x0001)
#define SNP24_ABORT_CAPTURE         ((Tparam) 0x0002)
```

becomes in Visual Basic

```
Const SNP24_START_AND_WAIT& = &H0
Const SNP24_START_AND_RETURN& = &H1
Const SNP24_ABORT_CAPTURE& = &H2
```

VISUAL BASIC FOR WINDOWS 95/98/NT

By making use of the Snapper ActiveX SDK, it is possible to write Visual Basic applications to control Snapper that run under Windows 95/98/NT. Refer to the ActiveX SDK or consult DataCell for further details..

EXAMPLE APPLICATIONS

Cross-platform capability

The example applications are supplied as demonstration source code to applications and libraries with the purpose of being compiled and capable of running on every one of the Snapper-supported operating systems. Use these as robust examples of how to use the Snapper for your particular purposes.

The applications support running in both console (i.e. text-only) and full-display modes, within the limits of the operating system in use.

The applications cover the Snp-24,Snp-8 and Snp-Dig16 cards and some hardware test utilities.

Snp24sq sequence mode video acquisition for area scan cameras,

Snp24nsq	non-sequence mode video acquisition for area scan cameras
D16asq	sequence mode video acquisition for digital area scan cameras
D16ansq	non-sequence mode video acquisition for digital area scan cameras
D15lsq	sequence mode video acquisition for digital line scan cameras
D16lnsq	non-sequence mode video acquisition for digital line-scan cameras
D16dsq	sequence mode video acquisition for digital data stream cameras
D16dnsq	non-sequence mode video acquisition for digital data stream cameras
Dmabench	measure the throughput data rate achieved with your Snapper and workstation.
Snapver	reports the Snapper library versions and driver interrupt in use.
Xamp	simple X-windows image viewer example for the TMG library.

The files:

Each example comes in its own directory with a set of makefiles, one for each OS. Use the relevant makefile and the instructions provided in the readme.txt file at the application root directory to build the application.