

Matrox

Genesis

Release Notes *for Genesis Native Library release 2.2*

M A T R O X
IMAGING

Contents

1.	<i>Overview</i>	5
2.	<i>HOST related topics</i>	6
2.1	Boards supported	6
2.2	Platform requirements	6
2.3	Genesis software installation	10
2.4	OS environment variables used	12
2.5	<i>GENESIS.INI file</i>	17
2.5.1	[GlobalSetting]	19
2.5.2	[EnvSetting]	27
2.5.3	[PrivateGlobalSetting]	35
2.5.4	[Node]	37
2.6	Directory structure	42
2.7	Description of software libraries	46
2.8	Notes regarding compilers	48
2.9	Setup compilers with <i>imcc</i>	50
3.	<i>LOCAL related topics</i>	51
3.1	TI TMS320C80 Software Tool Kit	51
3.2	Description of software libraries	54
3.3	Notes regarding compilers	56
3.4	Setup compilers with <i>imcc</i>	56

3.5	How to debug LOCAL code	57
3.5.1	Best time to start emulator	57
3.5.2	Best time to load LOCAL symbolic information	60
3.5.3	What happens to the HOST when LOCAL code is halted?	60
3.5.4	What happens to LOCAL when HOST restarts LOCAL ?	61
3.5.5	HOST messages when download in progress	62
3.5.6	Typical debugging session steps	64
4.	<i>imcc, imlk and imcl tools</i>	65
4.1	Compiler tool: imcc	65
4.2	Linker tool: imlk	69
4.3	Compile/Link tool: imcl	70
5.	<i>Running a demo</i>	72
6.	<i>Appendix</i>	76
6.1	IMCC.BAT reference	76
6.2	IMLK.BAT reference	86
6.3	Default <i>genesis.ini</i> file	90

1. Overview

Four sections comprise this document: HOST (PC environment), LOCAL (code running on the TMS320C80 or 'C80), compile and link tools, and how to run a basic demo.

The HOST section describes the following: HOST requirements, software installation, what this release includes, compile and link tools, and how to run the basic demo.

The LOCAL section, for DTK users, includes: comments on the TI TMS320C80 Tools Kit, and describes how to debug LOCAL code with the TI TMS320C80 Emulator.

The compile and link tools section describe command tools provided with this release to be able to compile and link an application.

The section on running a demo presents the steps to follow to run a basic demo.

In addition to this document, we recommend that you consult file *[drive]:\genesis\doc\readme.txt* for the latest notes regarding this and future releases. This file contains a description of new features, new functionality, bug fixes, errata on documentation, etc.

2. HOST related topics

This section describes in detail aspects related to the HOST platform.

2.1 Boards supported

The Genesis Native Library currently support:

- Genesis main board with display with/without a grab module
- Genesis PRO board with one or two processing node.
- Genesis LC
- Meteor II/DIG
- DAC (Dual Asynchronous Channel) acquisition board for Genesis LC board and Genesis main board.

2.2 Platform requirements

To be able to run a Genesis Native Library application, the basic system features required are:

DOS:

- Watcom 11.0b is required for DOS applications.
- For DOS applications: DPMI services need to be available. 386MAX from Qualitas has a good DPMI server but you do not need to use this specific one; EMM386 from DOS will do the job (the one available from DOS 6.22 seems more stable than the one from DOS 6.20). The Genesis Native Library in Watcom - DOS/4G assumes there is a DPMI environment present when application is executed. When HIMEM.SYS is used alone, up to the total of

available “high” memory can be used for DMA transfers to or from the HOST. When HIMEM.SYS + EMM386.EXE + NOVCPI switch is used, only 12 Mbytes of 32 Mbytes of system memory can be used for DMA transfers to or from the HOST. **Never use HIMEM.SYS + EMM386.EXE without the NOVCPI switch** since 4K pages are permuted, so not a physical linear address. At least 2048 bytes of environment space is required to setup various variables used by *imcc* and *imlk*. Also, ANSI.SYS in CONFIG.SYS is required to properly run *genpivx.exe* and *geninter.exe*.

- Pure DOS (16-bit) applications will never be possible.
- A DOS application created with Watcom 11.0b won't run in a Windows NT command prompt, you must be in pure DOS mode to do so.

QNX:

- Watcom 10.6 for QNX is required.
- QNX 4.24 or 4.25
- Optionally Photon 1.12 or 1.13 as GUI
- The executables must be run as root. See install.txt for more details.
- No utilities found on Windows NT are available.
- *imcc* and *imlk* are not available.
- A default makefile is provided as an example.

Windows NT:

- Microsoft Visual C++ 6.0 (4.2 and 5.0 are compatible) is required for Windows NT 4.0 for *Intel x86* architecture.
- For Windows NT applications: Windows NT 4.0 Service Pack 3 and later, for *Intel x86* architecture, is required.
- If the Windows NT TI TMS320C80 Code Generation Tools are used, Texas Instruments recommends as a minimum requirement a Pentium class processor running at 90 MHz.
- Under Windows NT 4.0 the amount of DMA memory that can be allocated in Non Paged Pool memory is restricted to approximately $((PC_memory_in_Meg - 4) * 0.4)$ which give 11.2 Meg on 32 Mbytes system and 24 Meg on a 64 Mbytes. Take note that each application and driver that are executed take some space in Non Paged Pool memory. So the theoretical maximum of Non Paged Pool memory that can be requested by Mtxdma? drivers can rarely be achieve. The installation software will try to determine the true maximum of Non Paged Pool memory. If your system behave abnormally after installing the Genesis Native Libraries software please reduce the amount of DMA memory. You may use the utility *mildrv.exe* (discussed later) to achieve this. This first operational mode was used in release prior to 2.0. The second operationnal mode of *mtxdma* permit a much larger block of DMA memory in fact most of the system memory may be reserved that way. The Genesis Native Library installer will now use this method automatically.

ALL PLATFORMS:

- All synchronous command; like `imSyncHost()`, `imBufInquire()`, etc. have an application time-out set to fifteen seconds. If required, that time-out can be controlled by `imAppControl()` or by **TimeOutCommunicationGet** in the *genesis.ini* file. Consult the Genesis Native Library for more details. When a time-out occurs the application is not stopped; it just warns the user that a response from the Genesis is taking longer than expected. This can happen if many asynchronous commands are queued followed by a synchronous command. In that case the synchronous command will complain about the time-out but it will resume its execution as soon as the synchronous command is completed.

2.3 Genesis software installation

Install as described in Chapter 3 of the Genesis Installation and Hardware Reference manual.

The INSTALL utility copies the required Genesis software to your hard disk¹. In addition:

- For DOS and optionally Windows NT it modifies your *autoexec.bat* file. It will add three environment variables: GENESIS, BATCH, and RAMDRIVE. It will modify the PATH setting to include some useful Genesis paths.
- For QNX the environment variable GENESIS must be added manually in the sysinit.[node] file and the Genesis driver must also be added manually to this file. Please follow install.txt for more details.

¹ Hard disk can also be a network drive.

- For Windows NT, install will copy in the directory *[drive]:\genesis\drivers\nt4.0\i386* the Genesis drivers named *genesis.sys* and *mtxdma.sys*, and a Windows NT registry database utility, *mildrv.exe*. This utility will be called during Genesis installation. It adds to the registry database the following information
HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\Genesis and in HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\Mtxdma?².
Genesis service is the board's device driver, while Mtxdma? is for various generic Matrox services required by the Genesis device driver. After installation, users may again use utility *mildrv.exe* to add or remove DMA memory from Windows NT resource.

The INSTALL utility will create all required directories and will copy the Genesis software (libraries, executables, demos, sources, and utilities) to these directories.

² Mtxdma0.sys, Mtxdma1.sys, Mtxdma2.sys, etc.

2.4 OS environment variables used

Genesis Native Library

The Genesis Native Library requires many controls. Depending on the application or specific requirement it is not always possible to control the behavior of the library with a simple function call. To give maximal control to the library, external settings are used. One way to inform the Genesis Native Library of users settings is to use environment variables. Another way is to specify all settings in an initialize file (*.ini*). The *.ini* used by the Genesis Native Library is named *genesis.ini*. This file contains settings for the library, default files to download, paths for various file extensions, default hardware settings, and a list of Genesis boards found in the system. An initialize file is used instead of the registry database because with an *.ini* we provide for the easy portability of the software on various OS platforms without major recoding.

The installation program will first copy in the *genesis.ini* in *Genesis_Directory\shell* directory. It will then add to your *autoexec.bat* an environment variable named GENESIS. This variable will be set to the path where the Genesis Native Library is installed. For example:

set GENESIS=c:\genesis	(for DOS and Windows NT)
export GENESIS=/genesis	(for QNX)

The Genesis Native Library including utilities, will use this environment variable to find where the Genesis Native Library is installed. This variable IS REQUIRED for proper operation of the board and software environment.

With the previous example the *genesis.ini* file is found in directory 'c:\genesis\shell'.

In some circumstances, it may be required to provide a variation of the default *genesis.ini*. To keep a valid copy of the *genesis.ini* file, users may provide a different path for the location of the *genesis.ini* file and then customize it at will. To override the default location set the following variable to your desired path and copy the default *genesis.ini* file to that location. For example:

```
set GENINIPATH=%windir%\Profiles\my_user_name
```

GENINIPATH is always considered before %GENESIS%\shell path for *genesis.ini* file location.

With the GENINIPATH variable, each user may have their own Genesis setup.

If the Genesis Native Library cannot locate a valid *genesis.ini* file it will fail to communicate with any Genesis board. The first thing to verify is whether your application failed to do an `imDevAlloc()`.

All other variables described later can be set in two ways: by the *genesis.ini* setting or by an explicitly defined environment variable. Precedence of one over the other is described later under key control **IgnoreEnvVar** of the *genesis.ini* file description.

To simplify the following text only DOS and Windows NT path syntax is used. For QNX, standard UNIX path must be used. For example:

```
DcfPath=/genesis/dcf:/genesis/dcf/dalsa:
```

Genesis Native Library Command Interpreter

The command interpreter uses three environment variables that define editor path, help path, and command list path. Examples:

```
set INTER_EDITOR=c:\dos\edit  
set INTER_HELP=%GENESIS%\inter\hlp  
set INTER_CMDLISTS=%GENESIS%\inter\cmdlists
```

IMCC & IMLK tools³

The next two variables are used by these tools and they have to be defined before IMCC or IMLK is called. Consult IMCC.HLP or the Appendix section of this document for a full description of these variables.

BATCH	Will point to IMCC release directory ⁴ : set BATCH=%GENESIS%\util\ ⁵
RAMDRIVE	Set ideally to a virtual drive letter: in config.sys declare a device for a ramdrive.sys set RAMDRIVE=v:\ or to a substituted drive: subst v: %GENESIS%\util\tmp set RAMDRIVE=v:\

³ IMCC & IMLK are code generation tools used internally at Matrox in order to compile and link sources for various compilers and installations. They are provided for convenience rather than necessity. See imcc.bat and imlk.bat reference sections for more information regarding these tools.

⁴ Warning: the directory path must not exceed the 128 character limit imposed under DOS (not applicable under Windows NT).

⁵ Trailing slash is mandatory for BATCH and RAMDRIVE.

When IMCC installs itself for a particular compiler it defines a list of environment variables. For some compilers, some variables are undefined. These variables are declared in files `%GENESIS%\util\imccfile*.set`:

PATH_PROJECT	PATH_COMPILER	PATH_MSTOOLS
PATH_HOME_DRIVE	PATH_PC_EMU	COMP_NAME
C_TOOLS	WATCOM	MSTOOLS
DOBUILD	INCLUDE	C_DIR
A_DIR	C_OPTION	ASM_OPTION
D_DIR	LIB	LIBDOS
LIBWIN	LIBNT	OBJDIR
RSPFILE	EDPATH	LNKFORMAT
CHECKINF	WD	DOS4GPATH
INIT	DEFAULTLNKLIST	

2.5 GENESIS.INI file

This section explicitly describes all key controls found in the *genesis.ini* file and some environment variables that may override settings found in the *genesis.ini* file.

Keys enclosed by square brackets [...] denote sections. Never comment these lines.

A line or the end of the line can be commented with the # sign. Default value for a line that has a string parameter is a NULL pointer when commented. Default value for a line with a number as a parameter value is 0 when commented.

A key not recognized by the Genesis Native Library is ignored.

All keys are not case sensitive.

Conditional execution can be added anywhere in *genesis.ini* file to comment or activate a portion of it. Syntax is:

```
!if 1      # Read next lines until !endif keyword
...
!endif    # Ends !if 1 condition
...
!if 0      # Ignore next lines until !entil keyword
...
!endif    # Ends !if 0 condition
```

Pre-defined conditions also exist for:

```
!if 430FX  
!if 430HX  
!if 430TX  
!if 430VX  
!if 440FX  
!if 440LX  
!if 450KX
```

An echo to console output can be placed anywhere in the *genesis.ini* file.
Syntax is:

```
@echo. My string to print
```

There is no replacement pattern available in *genesis.ini* file like %...% associated with DOS batch file or \$(...) used by *imcc.bat* in *.set files. All paths must be explicit. The installation program will update the *genesis.ini* file according to the default Genesis path selection.

All examples presented here will assume the Genesis installation directory to be c:\genesis.

2.5.1 [GlobalSetting]

This section is used to set global settings of the Genesis Native Library. Keys found in this section cannot be overridden by an environment variable. For a list of default settings consult the *genesis.ini*, section found in the appendix. Keys are:

IgnoreEnvVar	When IgnoreEnvVar is set to 'NO' Genesis Native Library will search first in environment variables and if not found in <i>genesis.ini</i> file. When IgnoreEnvVar is set to 'YES' Genesis Native Library will search first in <i>genesis.ini</i> file and if not found or commented, in environment variables.
ApplicationMessaging	To control Genesis Native Library internal print. Under NT all internal prints are piped to GenCout. Under DOS there is no piping. Valid values are: _IM_NO_PIPE No printout. _IM_DEFAULT_PIPE Print in Pipe (according to GenCout under NT). _IM_KEEP_PIPE Under NT keep GenCout open for debugging purpose (or according to GenCout setting)

AutoRescanIni

When set to 'YES' Genesis Native Library will always update *genesis.ini* if hardware profile changes. If for any reason you need to maintain what is currently defined then set to 'NO'.

WARNING: Regardless of this setting, if you add or remove a back plane or cable from the VM Channel, the GRAB port 2, or the GRAB port 1, the software will not detect those changes and may report wrong information. To force an update of the *.ini* file when one of these interconnections is altered it is recommended to delete all [Node] sections from your *genesis.ini* file.

MaxVmDevicesAt33

When the *genesis.ini* is created at the first *imDevAlloc()* the VM Channel interconnection is analyzed. If there is more VM Channel devices than **MaxVmDevicesAt33** than the software will limit operation at 25 MHz even if **VmSpeedLimit** is set at 33. If **VmSpeedLimit** is already at 25 than value of **MaxVmDevicesAt33** is irrelevant. Running VM Channel at 33 MHz for more VM Channel devices than the default values may not give the proper result.

VmSpeedLimit

VM Channel interface can operate at 25 MHz with a flat cable or at 33 MHz with the backplane. However in some circumstances you may need to override the maximum clock speed. This feature is mostly useful when the long VM Channel backplane is used. 25 MHz is always selected for a flat cable. Speed selection is done at the first imDevAlloc() after a power ON / or system reset. If you change this setting you will have to reboot your system to make it active since the VM Channel interface is initialized only once.

DisplayBoardInfo

If set to 'YES' will display information on the board that is currently downloaded: board interconnections, board revision, board modification, manufacturing information, etc. Note that some variations of the Genesis family (Genesis Processor board for example) may not provide this information. If set to 'NO' will skip that display.

CommunicationDebugMode When debugging your application it may be important to stop when a synchronous function does not end normally. In that case set that field to 'YES'. In release mode, looping for the user to end the execution of an endless operation is not really useful. In that case set that field to 'NO'. The synchronous operation will end and will log an application error that can be retrieved with `imAppCatchError()`. Set to 'TRACE' to record start and stop time of command execution. On a timeout the file specified by **DiagDumpFile** is created. Later user may analyse execution trace with GenAnalyse utility (WinNT only). Note that when this control is set to 'TRACE' GenSnoop and GenPerf can't be used.

LargestDmaBlock The Genesis Native Library work in conjunction with the *mtxdma.sys* driver. This driver can map very large physical pages of memory. Getting a virtual pointer to a large area of physical memory may not be possible if virtual address space of the process is fragmented. Instead the Genesis Native Library will allocate a list of virtual pointers that map to that single large physical memory area. Each virtual pointer will map no more than **LargestDmaBlock**. This value must be a multiple of 2^n bytes and the smallest value is 4096 bytes.

- DisableDefaultMappingOfVirtualBuffer** If set to **yes** all space request by *mtxdma.sys* driver are not systematically map by *imDevAlloc()*. The virtual mapping is done only on allocation of a HOST buffer. Allocation of buffers will be slower.
- ReserveHostSpaceForLcBuffers** Some HOST memory is required to store buffer structure. This memory is allocated in Non Paged Pool Memory so a full Genesis can reference it when doing cross node operations. By default 64 Kbytes is reserved. This give more than 512 buffers. You can increase that value as long as there is enough space in MtxDma drivers.
- SizeOfTmpBufferForHostGrab** Temporary buffers in on board memory are required to achieve grab to HOST memory. By default 16Kbytes for temporary buffer is sufficient. However sometimes larger buffers may prevent grab overruns. You can consult document grabbing_from_genesis.pdf for full details on this issue.

InterruptDrivenOSB

When set to **yes**, force new OSB signaling method. This involve sending interrupt when there is an OSB update state. This method is more efficient since the CPU waiting on the OSB is at 100 % asleep. Set to **no** for default OSB signaling. The default one is a polling/yielding approach slightly less efficient than the interrupt driven OSB method.

TimeOutCommunicationGet

With this key you can control the timeout on synchronous commands. It is express in seconds. To force a very large timeout don't use IM_INFINITE but put a very large value (up to $2^{32} - 1$ seconds can be specified). It as the same effect as `imAppControl(IM_APP_TIMEOUT, ?)`. A call to `imAppControl()` will override this default setting.

LcTimerScale

This field specify the internal timer scale, in mSec, for Genesis LC and Meteor II/Dig. This is not the OS time slice. The internal timer scale is used for timeout on OSB for example. A small **LcTimerScale** will result in unnecessary system CPU usage. We recommand a large value like the default (100 mSec) or greater.

DisableStdGenesisMutex

Genesis LC/DAC specific. Don't use except under Matrox recommendation.

XferTimeOut

Used to validate if a time out occur during a PCI or VM transfer. A value of 0 deactivate the mode. A value different than 0 express the time out in seconds to use. 5 seconds is a valid time out for most PCI or VM transfer. Don't use except under Matrox recommendation.

2.5.2 [EnvSetting]

This section is used to provide default path and file names for the Genesis Native Library. Keys found in this section can be overridden by an environment variable. Refer to the **IgnoreEnvVar** key to see whether the value from the environment variables or the *genesis.ini* file will be used.

CoffPath

For the `imDevAlloc()` function, the third parameter is the Shell file string. If `ShellFile` is not NULL, `imDevAlloc()` will use that file name. If no absolute path⁶ is used in the `ShellFile` string as in “*shell.out*”⁷ then `imDevAlloc()` will first check in the current directory for the file. Otherwise `imDevAlloc()` searches in the path specified by the environment variable **CoffPath**. When an absolute path is used in the Shell file name **CoffPath** is irrelevant. A valid setting for **CoffPath** is:

`CoffPath=c:\genesis\shell;c:\my_dir\shell;...`

ElfPath

Same as **CoffPath**. It apply to ELF file.

⁶ Absolute paths are, for example: `.\shell.out`, `..\shell.out`, `\shell.out`, `[drive]:\shell.out`, `[drive]:shell.out`, `./shell.out`, `../shell.out`, `/shell.out`, `~/shell.out`

⁷ Extension `.out` is not required because `imDevAlloc()` assumes it is `.out`

CoffName

When ShellFile is NULL, imDevAlloc() gets the Shell file defined by **CoffName**. Again if no absolute path is used in **CoffName** environment variable (as in “*shell.out*”) then imDevAlloc() will first check in the current directory for the file. Otherwise imDevAlloc() searches in the path specified by the environment variable **CoffName**. When an absolute path is used in **CoffName** then **CoffPath** is irrelevant.

CoffName=shell.out

ElfName

Same as **CoffName**. It apply to ELF file.

CoffLoadOption

The behavior of the COFF/ELF Loader (or Shell Loader) can be altered by the environment variable **CoffLoadOption**. **CoffLoadOption** overrides options used by default by the C-Binding. Options are:

- S: Skip download verification
- B: Download .BSS section
- P: Print COFF file information on sections size
- F: Force download regardless of signature
- I: Ignore local CPU even if there is one present
- Q: Quiet. Do not print downloaded file header
- M: Mute all print messages
- R: Reset Local Shell regardless of the download
- W: Wait for the user to press a key before resuming operation
- E: Specify the number of entries⁸ in the communication buffer. Letter 'E' must be followed by a number (e.g., 'E512').

⁸ By default *shell.out* has a circular command list of 256 entries.

To activate an option add a '+' after the letter option. To deactivate an option add a '-' after the letter option.

Examples:

```
CoffLoadOption=S+ Q+ W-  
CoffLoadOption =P- M+  
CoffLoadOption =F+  
CoffLoadOption =F+ E512
```

Current default option is: None (Print downloaded Shell file header only).

FpgaPath

This is the path where imDevAlloc() can find appropriate FPGA file for the specific hardware of that node (see [Node] section for more details on FPGAs).

TifPath

imBufLoad() and imBufRestore() load TIFF and MIM files. Instead of hard coding an absolute path in code (demos, examples, interpreter command list, etc.) it is recommended to pass only the TIFF or MIM file name, such as in “*baboon.tif*”⁹. imBufLoad() will automatically search first in the current directory and then in the search path specified by **TifPath**. Example:

TifPath=c:\genesis\image;c:\my_img;...

MimPath

Same description as **TifPath**.

JpgPath

Instead of hard coding an absolute path in code (demos, examples, interpreter command list, etc.) it is recommended to pass only the JPG file name, such as in “*baboon.jpg*”. imJpeg...() will automatically search first in the current directory and then in the search path specified by **JpgPath**. Example:

JpgPath=c:\genesis\image;c:\my_img;...

⁹ Extension .tif is not required because imBufLoad() assumes it is .tif

DcfPath

imCamAlloc() is used to allocate cameras and load camera definitions. The second parameter is CamFile; the name of the camera definition file. If no absolute path is used in the CamFile string as in “rs170.dcf”¹⁰ then imCamAlloc() will first check in the current directory for the file. Otherwise imCamAlloc() searches in the path specified by the environment variable **DcfPath**. When an absolute path is used in the CAM file name **DcfPath** is irrelevant. A valid setting for **DcfPath** is:

DcfPath=c:\genesis\dcf;c:\my_dir\dcf;...

DcfName

When CamFile is NULL, imCamAlloc() gets the DCF file defined by **DcfName**. Again if no absolute path is used in the **DcfName** environment variable, as in “rs170o.dcf”, then imCamAlloc() will first check in the current directory for the file. Otherwise imCamAlloc() searches in the path specified by the environment variable **DcfPath**. When an absolute path is used in **DcfName** then **DcfPath** is irrelevant:

DcfName=rs170.dcf (or any valid DCF)

¹⁰ Extension .dcf is not required because imCamAlloc() assumes it is .dcf

VcfPath

imDispAlloc() is used to allocate the display and load display definitions when in “dual screen”. The fourth parameter is DispFile; the name of the video definition file. If no absolute path is used in the DispFile string as in “*vm105_60.vcf*”¹¹ then imDispAlloc() will first check in the current directory for the file. Otherwise imDispAlloc() searches in the path specified by the environment variable **VcfPath**. When an absolute path is used in the VCF file name **VcfPath** is irrelevant. A valid setting for **VcfPath** is:

VcfPath =c:\genesis\vcf

VcfName

When DispFile is NULL, imDispAlloc() gets the VCF file defined by **VcfName**. Again if no absolute path is used in the **VcfName** environment variable, as in “*vm105_60.vcf*”, then imDispAlloc() will first check in the current directory for the file. Otherwise imDispAlloc() searches in the path specified by the environment variable **VcfPath**. When an absolute path is used in **VcfName** then **VcfPath** is irrelevant:

VcfName =vm105_60.vcf (or any valid VCF)
--

¹¹ Extension .vcf is not required because imDispAlloc() assumes it is .vcf

DiagDumpFile

It sometime may happen that the Genesis Native Library loses contact or becomes out of sync with the Genesis board. This is known as a communication time out. In this situation it is hard to figure out what commands were received when the situation occurred. If the user provides a diagnostic dump file like *diagdump.gsp* the Genesis Native Library will dump the communication buffer state as soon as the communication time out occurs. Users can then analyze the history of the commands received by using the GenAna tool, available for Windows NT. There is no DOS viewer available.

2.5.3 [PrivateGlobalSetting]

Keys defined in that section are used to configure the Genesis Board. **Do not remove or change any of these keys without consulting with Matrox.**

Syntax of these keys is:

```
[device_name.]register_name.file_name=value
```

Where device_name. can be:

- 1- Absent when it represents any primary VIA, display VIA, or the PCI-TO-PCI Bridge, if the register_name exists for these devices.
- 2- primary. to explicitly initialize the register field of the primary VIA.
- 3- display. to explicitly initialize the register field of the display VIA.
- 4- ppb. to explicitly initialize the register field of the PCI-TO-PCI Bridge.

Advance feature:

With the *genesis.ini* file it is possible to program or read any PCI configuration space of any PCI device.

In the following syntax:

pci	Is mandatory at the beginning of the line
DevId/VendId	Is the Device + the Vendor Id of the PCI device to access.
RegOffset	Is the register offset in bytes in the PCI configuration space.
BitMask	Is a group of successive bits that represent the Bit mask of the field to access.
FieldValue	Is the value to be written at the BitMask .
RegValue	Is the Double Word value to be written at the RegOffset .

If line is equal to:

- pci.DevId/VendId.RegOffset.BitMask=FieldValue or
pci.0x122d8086.0x40.0x00007000=0x3
A bit field access is executed to that register. Value is written in field marked by the **BitMask**. Register offset is rounded to Double Word offset.
- pci.DevId/VendId.RegOffset=RegValue or
pci.0x122d8086.0x44=0x12345678
Value is written in the Double Word pointed to by the register offset. Register offset is rounded to Double Word offset. Register offset is rounded to Double Word offset.
- pci.DevId/VendId.RegOffset or
pci.0x122d8086.0x44
Register is read and then printed at the console.

- pci.DevId/VendId or
pci.0x122d8086
If device is found it is printed at the console

If the device does not exist the command is skipped. All devices that respond to the same Device and Vendor Id are written or read the same way.

WARNING: Please contact Matrox before making any changes using this method.

2.5.4 [Node]

The purpose of the **Node** section is to describe each individual node found in the Host system. It will also describe interconnections found between each node.

Values defined in **GlobalSetting**, **EnvSetting**, and **PrivateGlobalSetting** apply for all nodes unless modified in each specific node.

For each node, add any field of **EnvSetting** or **PrivateGlobalSetting** to configure that specific node. Prefixes primary. or display. can be added to any element of **PrivateGlobalSetting**. Each VIA can be configured individually. When those fields are not preceded by primary. or display., then it applies for both VIAs. When preceded by ppb. it applies to the PCI-to-PCI Bridge.

WARNING: Node(s) section(s) is(are) configured dynamically by first imDevAlloc() after a hardware change in the system (Genesis board(s) added or removed). Any field of **EnvSetting** or **PrivateGlobalSetting** set in the **Node** section is lost after Node section(s) is(are) updated. Only the **Node** section is affected by this, other sections are preserved.

NodeId

The **NodeId** is an Id composed by the System and Node parameters passed to imDevAlloc().

The equation for the **NodeId** is:

$NodeId = (((System + 1) << 4) + (Node + 1)).$

System and Node are printed as comment after the NodeId setting. If the NodeId needs to be changed there is no need to change the comment on the System and/or Node since the Genesis Native Library ignores them. Take note that all Ids allocated with imDevAlloc(), imThrAlloc, imBufAlloc(), imCamAlloc(), imDispAlloc(), etc contain that **NodeId** in the Id returned by those functions.

ProductId

This info field tell what product that node reference. Can have the following value:

IM_DEV_GENESIS,
IM_DEV_GENESIS_PPC,
IM_DEV_GENESIS_PRO,
IM_DEV_GENESIS_PRO_PPC,
IM_DEV_GENESIS_LC,
IM_DEV_GENESIS_MEMORY,
IM_DEV_METEOR_IL_DIG,
IM_DEV_DAC,
IM_DEV_GENESIS_UNDEF.

SlotNbr

The physical slot number where the Genesis board is located.

DeviceNbr	Is the pseudo PCI device number of that node. This device number is composed of: (primaryBusNb << 16) + (PpbDeviceNb << 8) + ViaDeviceNb. This device number will be reused in GrabPortConnection , VmPortConnection , and RasterBlasterConnection to define board interconnection.
CpuPresent	Which local CPU is used. C80 when a 'C80 is present or NO when no local CPU.
DisplayMode	Can have the following value: IM_SINGLE_SCREEN, IM_DUAL_SCREEN, IM_DUAL_HEAD, IM_NONE
VmBusSpeed	Can be set to: 25 or 33
GrabModulePresent	Whether the GRAB module is present (YES) or not (NO).
ExtModuleType	This info field tell what external device is attached to that board. Can have the following value: IM_EXTDEV_NONE, IM_EXTDEV_GENESIS_GRAB, IM_EXTDEV_METEOR_II_DIG, IM_EXTDEV_METEOR_II_DIG_MJPEG, IM_EXTDEV_DAC_R0, IM_EXTDEV_DAC_R1, IM_EXTDEV_DAC_R2, IM_EXTDEV_RASTER_BLASTER.

FpgaName

Interface to external devices of Genesis boards (main board with a grab module, Genesis PRO, Genesis LC, Meteor II) is done with an FPGA (Field Programmable Gate Array) on that board. This FPGA has to be programmed properly during `imDevAlloc()`. Instead of hard coding in the Genesis Native Library the file name associated to that FPGA, `imDevAlloc()` will search in **FpgaPath** for the file specified by **FpgaName**. **FpgaName** may also be an absolute path with the file name.

FpgaName =gengrab1.res or FpgaName =dac.res or FpgaName =mach_iv.res FpgaName =metiidig.res or FpgaName =rblaster.res
--

BufferPitchAlignment

By default buffers allocated will have a pitch a multiple of **BufferPitchAlignment**. By default this value is 4 bytes. However when Genesis with PPC is present the default will be 16 bytes for all nodes so PPC can access appropriately C80/LC/HOST buffers. In case there is an incompatibility with a C80 application user may set **BufferPitchAlignment** back to it's default value of 4 bytes.

GrabPortConnection	Is a list of DeviceNbr attached together in the form: 0x001004,0x000d04,... No connection when the field is set to 0.
VmPortConnection	Is a list of DeviceNbr attached together in the form: 0x001004,0x000d04,... No connection when the field is set to 0.
VmExternalDeviceId	<p>If an external VM Channel device is added in the VM Channel chain, set the Device Id associated to that device so that Genesis Native Library will be able to chain this Device with Genesis device(s). More than one device can be added in the chain. For example: 15 (for 1 external device with Device Id of 15) or 15,14,13 (for 3 external devices). No external device when the field is set to 0. Nodes in the same System must have identical VmExternalDeviceId information. There is no automatic recognition of non-Genesis VM device. Users will have to manually modify this key to reflect their hardware.</p>
RasterBlasterConnection	Is a list of DeviceNbr attached together in the form: 0x001004,0x000d04,... No connection when the field is set to 0.

2.6 Directory structure

Directories are structured so that more can be added in the future without affecting the current structure. Future add-ons will be: new HOST and LOCAL examples, new headers, new documentation, etc. Take note that this structure will certainly evolve in the future but what is currently defined will stay. Directories ending with ??? designate new directories that may appear in the future. The following is a brief descriptions of each directory.

<code>%GENESIS%\dcf</code>	Genesis DCF for standard acquisition board
<code>%GENESIS%\dcf\camera_brand</code>	Genesis DCF for various manufacturers
<code>%GENESIS%\dcf\custom</code>	DCFs for Matrox Private customers. These DCFs were designed for proprietary cameras or scanners. Matrox assigns a name to each custom DCF. Future releases will always maintain the naming for Private customers.
<code>%GENESIS%\dcf.mac</code>	Genesis DCF for Multiple Asynchronous Channel Acquisition Board (MACAB)
<code>%GENESIS%\dcf.mac\camera_brand</code>	DAC and MACH_IV Genesis DCF for various manufacturers
<code>%GENESIS%\dcf.mac\custom</code>	DAC and MACH_IV DCFs for Matrox Private customers. These DCFs were designed for proprietary cameras or scanners. Matrox assigns a name to each custom DCF. Future releases will always maintain the naming for Private customers.
<code>%GENESIS%\demo</code>	Demos in executable form for DOS
<code>%GENESIS%\demo\dos</code>	Demos in executable form for Windows NT
<code>%GENESIS%\demo\winnt</code>	Logging directory (compiled output)
<code>%GENESIS%\demo\log</code>	Contain 'makefile' file for sources in \demo\
<code>%GENESIS%\demo\make</code>	
<code>%GENESIS%\demo\obj</code>	Stack objects for sources in \demo\ for Watcom
<code>%GENESIS%\demo\obj\doswat</code>	Register objects for sources in \demo\ for Watcom
<code>%GENESIS%\demo\obj\doswatr</code>	Objects for sources in \demo\ for Visual C++
<code>%GENESIS%\demo\obj\nt</code>	Future compiler objects directories
<code>%GENESIS%\demo\obj\???</code>	Demo source for play.exe
<code>%GENESIS%\demo\play</code>	

%GENESIS%\demo\???

Other future demo

%GENESIS%\doc

Documentation, release notes, etc.

%GENESIS%\drivers

Various drivers: NT, Win95, etc.

%GENESIS%\drivers\nt3.51

Registry database utility and Genesis drivers

%GENESIS%\drivers\nt3.51\i386

For Intel 386 platform

%GENESIS%\drivers\nt4.0

Registry database utility and Genesis drivers

%GENESIS%\drivers\nt4.0\i386

For Intel 386 platform

%GENESIS%\examples

Various HOST examples

%GENESIS%\examples\host

Example executables for DOS

%GENESIS%\examples\host\dos

Example executables for Windows NT

%GENESIS%\examples\host\winnt

%GENESIS%\examples\host\dual

Dual Node example

%GENESIS%\examples\host\log

%GENESIS%\examples\host\make

%GENESIS%\examples\host\misc

Some examples

%GENESIS%\examples\host\obj

%GENESIS%\examples\host\obj\doswat

%GENESIS%\examples\host\obj\doswatr

%GENESIS%\examples\host\obj\nt

%GENESIS%\examples\host\obj\???

%GENESIS%\examples\host\thresh

Threshold example (for DTK users)

%GENESIS%\examples\host\???

%GENESIS%\examples\local

Various LOCAL examples (for DTK users)

%GENESIS%\examples\local\log

%GENESIS%\examples\local\make

%GENESIS%\examples\local\obj

%GENESIS%\examples\local\obj\little

%GENESIS%\examples\local\obj\???

%GENESIS%\examples\local\thresh

Threshold example (uses PPs)

%GENESIS%\examples\local\???

%GENESIS%\image

Various images in TIFF, MIM, ... format

%GENESIS%\inter

Interpreter basic command list

%GENESIS%\inter\cmdlists

Interpreter examples

%GENESIS%\inter\examples

Interpreter help files

%GENESIS%\inter\hlp

<code>%GENESIS%\lib</code>	Object libraries for all platforms
<code>%GENESIS%\lib\obj</code>	
<code>%GENESIS%\lib\obj\doswat</code>	HOST C-Binding libraries for Watcom (stack version)
<code>%GENESIS%\lib\obj\doswatr</code>	Watcom (register version)
<code>%GENESIS%\lib\obj\nt</code>	HOST C-Binding libraries and DLLs for Visual C++
<code>%GENESIS%\lib\obj\little</code>	LITTLE endian LOCAL libraries and objects
<code>%GENESIS%\lib\obj\???</code>	Other platforms
<code>%GENESIS%\patch</code>	Patch to apply to sources or libraries
<code>%GENESIS%\patch\wstub</code>	Watcom stub for DOS4GW and DOS4GW
Professional	
<code>%GENESIS%\shell</code>	‘C80 COFF file Shell
<code>%GENESIS%\src</code>	Genesis Native Library Source
<code>%GENESIS%\src\headers</code>	Genesis Native Library headers
<code>%GENESIS%\src\headers\common</code>	Common to both HOST and LOCAL
<code>%GENESIS%\src\headers\host</code>	
<code>%GENESIS%\src\headers\local</code>	For DTK users.
<code>%GENESIS%\src\???</code>	Eventually C-Binding sources, etc. (for DTK users)
<code>%GENESIS%\usr</code>	User directory (used by <i>imcc</i> when compiling for ‘C80 under UNIX)
<code>%GENESIS%\usr\???</code> ¹²	Any valid user name (see USER in .cshrc)
<code>%GENESIS%\util</code>	Matrox universal compile and link tools
<code>%GENESIS%\util\dos</code>	Genesis DOS utilities
<code>%GENESIS%\util\emu</code>	PDM script for DTK users
<code>%GENESIS%\util\winnt</code>	Genesis Windows NT utilities
<code>%GENESIS%\util\winnt\ramdisk</code>	Ramdisk driver for Windows NT
<code>%GENESIS%\util\imccfile</code>	Response file and setting for compilers

¹² If you intend to compile/link with *imcc/imlk* tools with the TI TMS320C80 Code Generation Tools for UNIX, you must create your user directory here. This directory name must be identical to your UNIX USER name. This directory will be used for temporary storage by the TI TMS320C80 Code Generation Tools. If more than one user is using the same Genesis Native Library install branch then creating for each user a directory in `%GENESIS%\usr` will ensure that no one clashes with other user’s temporary files.

%GENESIS%\util\imccrun
%GENESIS%\util\imccrun.nt
%GENESIS%\util\lnklist
%GENESIS%\util\tmp

Executable for tools
Executable for tools for Windows NT
Default '*imlk*' link list
RAMDRIVE directory if substituted drive is used

%GENESIS%\vcf

Genesis VCF

2.7 Description of software libraries

To link a HOST C-Binding application, include the following libraries:

- For Watcom¹³:

bufcomm.lib	Buffers functions
cbind.lib	C-Binding functions
gengrab.lib	Standard acquisition board functions
hlcomm.lib	Common high level functions
ldshell.lib	Shell loader
lcomm.lib	Common low level functions
llhost.lib	HOST low level functions
macab.lib	Asynchronous acquisition board functions
multiff.lib	MIL Tiff loader functions
mp_task.lib	Tasking functions
mp_timer.lib	Timer functions
- For Microsoft Visual C++ with all objects linked in the application (larger executable but no DLL to provide):

bufcomm.lib	Buffers functions
cbind.lib	C-Binding functions
gengrab.lib	Standard acquisition board functions
hlcomm.lib	Common high level functions
ldshell.lib	Shell loader
lcomm.lib	Common low level functions
llhost.lib	HOST low level functions
macab.lib	Asynchronous acquisition board functions

¹³ From %GENESIS%\lib\obj\doswat for the stack parameter passing calling convention and %GENESIS%\lib\obj\doswatr for the register parameter passing calling convention.

miltiff.lib ¹⁴	MIL Tiff loader functions
mp_task.lib	Tasking functions
mp_timer.lib	Timer functions

- For Microsoft Visual C++ with objects in the DLL (the application may not necessarily need to be recompiled when a new *genesis.lib* is provided):

genesis.lib ¹⁵	Genesis import library
miltiff.lib	MIL Tiff loader functions

To simplify linking '*imlk*' is useful. '*imlk*' has universal link definitions for various compilers. Consult the 'Linker tool: imlk' section for more details. The library list for Watcom is in file %GENESIS%\util\lnklist\host.lst. The library list for Microsoft Visual C++ is in file %GENESIS%\util\lnklist\hostnt.lst. Be sure to use the appropriate Watcom libraries when linking (register passing convention is not compatible with stack convention). For Watcom see file %GENESIS%\util\lnklist\play.lnk for a stand alone link list description (was created by *imlk*) and %GENESIS%\util\lnklist\playnt.lnk for Microsoft Visual C++.

¹⁴ Combine with miltiff.dll.

¹⁵ Combine with genlib.dll

2.8 Notes regarding compilers

All C-Binding API functions are C like functions, but all sources were compiled in C++. It was done so because C++ reinforces C syntax checking and prototypes. Some portion of the Genesis Native Library is built with C++ construction.

For Windows NT, if you intend to use the Microsoft Visual C++ IDE, Genesis tools such as *imcc* and *imlk* are useless. However we do not provide a default project to configure the IDE. You will have to follow these steps to properly configure a project for Genesis.

- In 'Tools/Options/Directories':
 - In 'Include files' add the following includes directories:
 - `%GENESIS%\src\headers\common`
 - `%GENESIS%\src\headers\host`
 - In 'Lib files' add the following libraries' directories:
 - `%GENESIS%\lib\obj\nt`
- In 'Build/Settings':
 - Choose C/C++ tab
 - Choose General in category
 - Select all Items in Settings for
 - Add in preprocessor definitions all defines found in `%GENESIS%\util\imccfile\msvc_response_file.rsp`¹⁶
 - Choose link tab
 - Choose General in category
 - Select all items in Settings for
 - Add genlib.lib in object/library modules
- End project definition by 'Project/Update All Dependencies'

¹⁶ msdevnt.rsp for Microsoft Visual C++ 6.0.

2.9 Setup compilers with *imcc*

WATCOM:

%GENESIS%\patch\wstub directory contains a Watcom Stub replacement for Watcom Tools that can work with both DOS4GW (provided with Watcom Tools) and DOS4GW Professional.

To setup a compiler environment, use:

imcc /c watcom	for DOS under DOS with stack convention
imcc /c watcomr	for DOS under DOS with register convention
imcc /c watcomn	for DOS under Windows NT with stack convention
imcc /c watcomnr	for DOS under Windows NT with register convention

Microsoft Visual C++:

To setup a compiler environment, use:

imcc /c msdevnt	Windows NT under Windows NT
imcc /c msdevdll	for Windows NT/DLL under Windows NT

3. LOCAL related topics

This section describes in detail LOCAL aspects. It is reserved to those who use the Genesis Developer's Tool Kit (DTK).

3.1 TI TMS320C80 Software Tool Kit

What follows is information in addition to the one provided in TI documentation.

TI TMS320C80 Code Generation Tools

Some of the 'C80 libraries were modified to fulfill Matrox requirements. We suggest that you always link your LOCAL shell with those modified libraries.

TI TMS320C80 On-line Reference

On-line reference can be easily installed on both Sparc and/or PC platforms. Just follow the TI instruction. Warning; if you choose to install (copy the CD ROM) to hard disk, we recommend installing it at the root of your drive. If for some reason this is not your choice, you may have to edit the file *iview.ini* in %windir% to correct the path for the on-line reference tools. Be sure also that TEMP and FULTEMP are set properly (see the section on HOST environment variables used).

TI TMS320C80 Emulator

The emulator executable has to be installed on the same Sparc station as the emulator itself. To install the emulator in a Sparc station you must be in super user mode.

To install PC emulator please follow emulator provider instructions.

If you try to start emulator immediately without proceeding with the initialize sequence of the Genesis, it will fail because the Genesis JTAG interface is not initialized yet. Consult the section ‘How to debug LOCAL code’ before using emulator debugger. However doing an emulator reset, *emurst*¹⁷, is possible in order to verify if the emulator is connected properly to the HOST system. In that case, Genesis does not need to be initialized.

On the Genesis main board, since there is only one JTAG device (the ‘C80) it is required to create a specific board.dat¹⁸ for the emulator. However the Genesis Processor board is equipped with one or two ‘C80s depending on the option installed. In this case you have to inform the emulator software how many JTAG devices there is in the chain. If two ‘C80s are present then a new board.dat that matches the chain is required.

For convenience, Matrox provides in %GENESIS%\util\emu some useful files that can be used to easily configure the emulator for the Genesis and the Genesis Processor boards.

¹⁷ Utility to reset emulator hardware. Available in TI TMS320C80 Emulator CD ROM.

¹⁸ See Texas Instruments documentation on the emulator to set that file.

Take note that there is no restriction opening two mpemu or ppemu for each node of the Genesis Processor board at the same time.

TI TMS320C80 Errata

Matrox suggest that you consult Texas Instrument WEB site at <http://www.ti.com> to obtain the latest errata on the 'C80 and 'C80 tools.

3.2 Description of software libraries

To link a LOCAL C-Binding application, include the following libraries for the TI TMS320C80 Tools

bufcomml.lib	Buffers functions
cbindl.lib	C-Binding functions
gengrabl.lib	Standard acquisition board functions
hlcomml.lib	Common high level functions
hllocall.lib	LOCAL high level functions
llcomml.lib	Common low level functions
lllocall.lib	LOCAL low level functions
macabl.lib	Asynchronous acquisition board functions
mp_taskl.lib	Tasking functions
mp_timel.lib	Timer functions
procbinl.lib	imBin...() functions
procbibl.lib	imBlob...() functions
procbufl.lib	imBuf...() functions
procftrl.lib	imFloat...() functions
proctgenl.lib	imGen...() functions
proctintl.lib	imInt...() functions
procjpgl.lib	imJpg...() functions
procpatl.lib	imPat...() functions
release.obj	COFF header release ID

To simplify linking '*imlk*' is useful. '*imlk*' has universal link definitions for various compilers. Consult the 'Linker tool: imlk' section for more details. The library list for the TI TMS320C80 Code Generation Tools is in file

`%GENESIS%\util\lnklist\local.lst`. LITTLE endian libraries with the form `lib_name1.lib` are found in `%GENESIS%\lib\obj\little`. In link list `%GENESIS%\util\lnklist\local.lst` only `lib_name.lib` are referenced. This is because the *imlk* tool handles the extra 'l' required for LITTLE endian libraries. If you choose to create your own link list and not use the *imlk* tool, then do not forget to add this extra 'l' to library names. Note also that *release.obj* is required in the link list to force a default COFF header. See file `%GENESIS%\util\lnklist\shell.lnk` for a stand alone link list description (was created by *imlk*).

3.3 Notes regarding compilers

Matrox only releases libraries and executables for the LITTLE endian compiler.

During compilation we use the switch `-mv` for the MP compiler. This switch is not documented in the TI TMS320C80 Code Generation Tools manual. This switch maintains the old *volatile* keyword meaning, which is: bypass cache when used (DEA - direct external access). From release 1.13 of the TI TMS320C80 Code Generation Tools the default meaning of this keyword is as defined by ANSI C: Do not optimize the source line when code optimization is active. The previous definition was: Do a DEA when *volatile* keyword is used. If your application does not require DEA to I/O resources¹⁹ then do not use the `-mv` switch. *imcc* tool automatically appends this switch to the *mpcl* command line.

3.4 Setup compilers with *imcc*

Install the TI 'C80 Tools as describe in TI install program. We recommend that you install TI 'C80 Tools in the `c:\c8xtools` directory.

To setup a compiler environment, use:

`imcc /c lmvnt` for 'little endian' under Windows NT

¹⁹ Which should be the case because Matrox provides all functions for I/O control.

3.5 How to debug LOCAL code

As no source code is provided for LOCAL code, all released libraries are compiled without symbolic information. This does not mean that it's impossible to debug custom code. Any custom LOCAL code can be compiled with symbolic information and later be linked with Genesis Native Libraries.

Despite the fact that an emulator can load a complete COFF file (the Shell) into the 'C80 system memory through the JTAG port, this feature is not supported with Genesis boards.

Therefore all code downloading will be done by the HOST, even though this excludes symbolic information. Symbolic information is loaded later by the emulator with the *sload* command.

3.5.1 Best time to start emulator

It is not possible to single step code from halt state (state present just after the reset). Reset time is critical and some portion of the code needs to be executed very quickly after 'C80 receives the reset.

To be able to determine the best time to start the emulator, you must first enable wait pause (W+) in the **CoffLoadOption** environment variable. At download time the HOST will at some point print "'C80 not started yet!" when wait is enabled. This is the proper time to start the emulator program if you wish.

When you see "'C80 not started yet!", the 'C80 was reset and unhalted by the COFF loader. The 'C80 is now looping in the MP (Master Processor) Genesis startup code. At this point the 'C80 is no

more in halt state and the C init code is not even executed yet. MP wait for the signal from the HOST to resume the startup code.

The emulator can be started at any time if LOCAL code is already started and is running normally (see section ‘What happens to HOST when LOCAL code is halted?’ for behavior of HOST code in this case).

Never use reset and/or restart command of the emulator. Doing so will break the connection between HOST and LOCAL. Remember that the MP or PPs are halted when the emulator program is invoked. Just starting the MP emulator is sufficient to stop the PPs. If you plan to debug only MP code it is a good practice to issue a runf (run free) command for each PP. Not doing so will prevent PPs from running normally after MP emulator is started. The instruction pointer (IP) points to the instruction line halted. To restart the LOCAL processor simply press F5.

WARNING: Because a board reset closes the connection of the JTAG interface, the emulator programs, *mpemu* or *ppemu*, cannot connect to the 'C80 if the Genesis was never initialized by a call to `imDevAlloc()`.

If you receive the following message

“CANNOT INITIALIZE THE TARGET !!

- Check I/O configuration
- Check cabling and target power”

when executing the *mpemu* or *ppemu* command, then you should consult the manual ‘TI TMS320C80 Workstation Emulator, Installation Guide’ section ‘Problems when invoking the debugger’ for possible explanations. Alternatively be sure that you issue a call to `imDevAlloc()` to initialize the Genesis system properly.

Note that in some rare circumstances, when debugging code (and all windows in emulator programs turn red), it may become impossible to reset the 'C80 properly by forcing a reset to it. In that case a power off of the system is recommended. Powering off the emulator box is sometimes also required.

After each debugging session it will be a good practice to always reset emulator with *emurst* utility to close JTAG connection with 'C80 since next `imDevAlloc()` will complain if JTAG is not free when code is downloaded.

3.5.2 Best time to load LOCAL symbolic information

When the emulator is started and a valid Shell is downloaded, symbolic information can be loaded by the emulator. To do this use the *sload* command in emulator:

```
sload shell.out
```

This command will only load symbolic information. If the LOCAL processor was running a valid Shell when the emulator was invoked, then the instruction pointer (IP) will point exactly where the emulator halted the LOCAL CPU. Resuming execution is done by pressing F5 to run the 'C80 master processor (MP).

After symbolic information is loaded, break points can be placed anywhere in the code.

3.5.3 What happens to the HOST when LOCAL code is halted?

Debugging LOCAL code involve HOST code too. There is some time-out mechanism in HOST code. These mechanisms are there to signal an error when a command take too long to resume its execution. When debugging LOCAL code user often stop execution of the 'C80. Since HOST code have no idea what's going on it may consider that as a time-out. Which is obviously not the case. To keep the HOST synchronize with LOCAL execution set **CommunicationDebugMode** key in *genesis.ini* file to YES.

If the HOST was exactly at a point where it was waiting for a reply from the LOCAL, HOST will print a message explaining that the HOST time-out has expired.

If no key is pressed, HOST will wait until LOCAL resumes its execution. The wait is visible as a turning prompt on the screen for DOS or a dialog box under Windows NT. At that moment HOST will check periodically for a reply from LOCAL.

HOST waiting is aborted when a key is pressed on the HOST system (the aborted command will fail, and later commands could be affected by this abort).

3.5.4 What happens to LOCAL when HOST restarts LOCAL ?

When debugging, it might be interesting to restart the HOST code without leaving the emulator and then reissue the `imDevAlloc()` and hopefully stop at previous break point set in the LOCAL code.

This scenario is however not possible. The function `imDevAlloc()` will not resume its execution until the emulator is disconnected (by leaving the emulator and then issuing an *emurst*) from the 'C80.

Again the reason for this, is the critical time required for the 'C80 reset sequence (it is not a software but a hardware issue).

3.5.5 HOST messages when download in progress

All HOST messages are mute by default during download. If M- and P+ are used the COFF loader will display downloading steps.

Typically the COFF loader will display the following information:

‘C80 JTAG Emulation is active.

or

‘C80 JTAG Emulation is active. Do an EMURESET now.

File header:

Number of sections =14

Number of entries in symbol table = 37480

File header flags = 0x1103

Entry point address = 0xffe000b4

Endians: HOST - Little LOCAL - Little

Downloaded file identification:

Matrox ‘C80 Development Board version 1.1

Copyright (C) 1997 Matrox Electronic Systems, Ltd.

All Rights Reserved.

Wed Apr 30 21 20:46:44 1997

Section .text (214516 bytes, starts at 0xFE000000)
downloaded

Section .ptext (182192 bytes, starts at 0xFE3B0000)
downloaded

...

...

“C80 JTAG Emulation is active” or “C80 JTAG Emulation is active. Do an EMURESET now” is printed when the emulator program is active. In the second case, because the HOST reset the ‘C80 it is better to quit the emulator, do an *emurst* on the emulator, and restart the emulator when the HOST finishes executing `imDevAlloc()`.

The remaining messages are printed to help identify the loading sequence.

3.5.6 Typical debugging session steps

- Configure COFFLOADOPTION with required switches
- Call imDevAlloc() to download and start LOCAL code
- COFF loader displays file information
- COFF loader displays: “C80 not started yet!”
- COFF loader displays: “?”
- *This is the best time to start the emulator and to load symbolic information. Break points can be set now or later if required*
- Start the MP code by pressing F5 or run.
- Pressing a key on HOST side will complete Genesis MP startup.
*If MP was not started previously by run you will have up to 2 seconds here to start MP code by pressing F5 or run. If run is not executed in time the HOST will print:
“C80 stalled or defective interrupts!”
In that case you have to restart the sequence*
- If startup is successful the HOST will print:
“C80 just started with Shell shell.out.”
- *If some breakpoints were set then the LOCAL will eventually break there if that code is executed*

4. *imcc*, *imlk* and *imcl* tools

A Ramdrive (or ramdisk) or a pseudo Ramdrive is required for three reason. First, in DOS there is a limit of 128 characters on a command line. In some situations we have to reference with an absolute path where temporary files are located; not using a short path as v:\ may make the command exceed the 128 characters limit. Second, we need a working version of all *imcc/imlk* files. Files in %GENESIS\util\ are not working version. Those files are transferred and modified in the Ramdrive to be used later by the working version of *imcc/imlk*. Third, if a real Ramdrive is used efficiency of the tools is noticeable for big projects.

4.1 Compiler tool: *imcc*

The '*imcc*' tool compiles any sources with a .c, .cpp, .ppc, .asm, and .s file extension. It can be configured to work with Industry standard compilers like: Borland, Microsoft, Watcom, etc., and even the TI TMS320C80 Code Generation tools. Compiler configuration is done by modifying .rsp, .set and .fmt files in the directory %GENESIS%\util\imccfile. Consult section '6.1 IMCC.BAT reference' item 12 for a QUICK START.

Compiler/linker selection is done with the following command:

`imcc /c the_compiler_name`

With '*imcc*', no makefile is required because it creates one automatically. For example, if a directory contains: file1.c, file2.c, file3.c, file4.s, file5.asm, then the command

```
imcc [/d]20 *
```

will compile all files in that directory regardless of the extension of the files. To compile only one file, use the following instruction:

```
imcc /d file2 (extension is never required)
```

Wildcards (* and ?) are accepted in file names.

If a directory structure is created according to the one that 'imcc' accepts then 'imcc' can build nested directories. The \make\ directory contains a *makefile*²¹ used by 'imcc' to build a group of source's directories. To build a group of directories use:

```
imcc [/d] *
```

in the directory that includes all sub-directories to build.

'imcc' options are generals and valid for all supported compilers.

.set files define environment variables used by the specific compiler. Length of variables is not a problem since 'imcc' manages them properly for various OS platforms (even DOS is not limited to 128 characters!). To ease porting to many locations i.e. any hard drive or any network, a mechanism is used when 'imcc' installs itself on the RAMDRIVE. In the original

²⁰ Consult imcc.hlp documentation for list of available options.

²¹ This *makefile* can be created automatically by a create make file utility named 'cmkfile'.

%GENESIS%\util\imccfile*.set files, all \$(path_something)²² are converted to what PATH_something is defined as in environment variable (if already defined), or obtained from what is defined in the .set file (ideally PATH_something definitions will always precede any other environment variable declarations in .set file).

.rsp files define compiler specific switches. These files normally should not be customized by anyone in order to insure compatibility with Matrox Genesis Libraries.

WARNING: .set, .rsp and even .fmt (see 'imlk' section description of .fmt) files may not correspond at all to your system setup. Matrox has no way to determine what is your preferred installation (e.g., hard disk or network installation) for compiler tools and cannot impose one. For this reason those files may have to be customized by users.

Remember to apply any modifications to files .set and .rsp in %GENESIS%\util\imccfile if you require a permanent modification.

All variables PATH_something (where something is anything) included in files %GENESIS%\util\imccfile*.set are used by imcc the first time its installed. PATH_something and \$(environment_variable) will be used by imcc to properly install a running version of imcc in the ramdrive. If required, %GENESIS%\util\imccfile*.set files can be configured to properly match custom setup. Just remember that any sub-string that contains \$(PATH_something) or \$(environment_variable) will be replaced by their equivalent definition.

²² Lower case \$(path_something) are converted to lower case path. Upper case \$(PATH_SOMETHING) are converted to UPPER CASE PATH.

Any environment variable can be used in .set files. The syntax is for example:

```
PATH_PROJECT=$(GENESIS)
```

where \$(GENESIS) will be replaced by the current value of the GENESIS environment variable. In the previous example if GENESIS is equal to c:\genesis then PATH_PROJECT will be equal to c:\genesis. Also all occurrences of \$(PATH_PROJECT) or \$(path_project) will be replaced by c:\genesis.

Here is an example of the original settings found in
%GENESIS%\util\imccfile*.set versus converted .set files found in
%RAMDRIVE%*.set:

%GENESIS%\util\imccfile*.set:

```
PATH_PROJECT=$(genesis)
PATH_COMPILER=$(msdevdir)
...
INCLUDE=$(path_project)\src\common;$(path_project)\src\host;$(path_compiler)\include;
```

%RAMDRIVE%*.set:

will be replace by:

```
PATH_PROJECT=c:\genesis
PATH_COMPILER=c:\msdev
...
INCLUDE=c:\genesis\src\common;c:\genesis\src\host;c:\msdev\include;
```

4.2 Linker tool: imlk

'*imlk*' links objects and libraries together. The link list is compiler independent. It can be configured to work with Industry standard compilers/linkers like: Microsoft, Watcom, etc., and even TI TMS320C80 Code Generation tools. Linker configuration is done by modifying .set and .fmt files in directory %GENESIS%\util\imccfile.

Read the notes and WARNING in 'Compiler tool: imcc' section.

Remember that '*imlk*' will operate for the compiler set by '*imcc*'. To set up for a specific compiler/linker use a command like:

```
imcc /c the_compiler_name
```

'*imlk*' options are general and valid for all compilers/linkers supported.

For example, to link a Watcom application do the following:

```
imcc /c watcom (if not already done)
imlk target /L your_link_list /L %GENESIS%\util\lnklist\host.lst [other_options]
```

For example, to link a Microsoft Visual C++ application do the following:

```
imcc /c msdevnt
imlk target /L your_link_list /L %GENESIS%\util\lnklist\hostnt.lst [other_options]
```

After compiler setup is done with '*imcc /c the_compiler_name*', you can link more generally by using:

```
imlk target /L your_link_list /L %DEFAULTLNKLIST%
```

where DEFAULTLNKLIST is an environment variable set by 'imcc /c *the_compiler_name*' that defines the default link list to use for that compiler selection.

your_link_list parameter can be a link list, an object or a library. Consult the *imlk.bat* reference for more details on the 'imlk' tool. File *doit.bat* and *doitnt.bat* in directory %GENESIS%\demo\play are examples of usage.

.fmt files define the linker link list format. Again each linker has its specific format. From the general link list format 'imlk' will be able to create a specific link list file for each linker. See files %GENESIS%\util\lnklist\play.lnk for Watcom, %GENESIS%\util\lnklist\playnt.lnk for Microsoft Visual C++, and %GENESIS%\util\lnklist\shell.lnk (for DTK users) for a stand alone link list description (were created by *imlk*).

4.3 Compile/Link tool: imcl

imcl.bat is a compile and link utility for a single source. It is a batch file that first calls *imcc.bat* to setup compiler selection: watcom for DOS and msdevnt for Windows NT. It compiles the file with symbolic information and it will put the object in the same directory as the source. Then a link is issued with *imlk.bat* to link an executable in the same directory as the source. Example of usage:

```
cd %GENESIS%\demo\play  
imcl play
```

After execution, you will find in %GENESIS%\demo\play directory, files play.log, play.obj, play.exe.

5. Running a demo

This release includes one demo of the current functionality of the Genesis Native Library. The demo source is in `%GENESIS%\demo\play` directory. A ready to run executable is available. The executable name is *play.exe* in `%GENESIS%\demo\dos` for DOS and in `%GENESIS%\demo\winnt` for Windows NT. These executables were compiled from `%GENESIS%\demo\play\play.c` with Watcom 11.0b and DOS4GW Professional from Tenberry/Rational Systems Inc. for the DOS version and Microsoft Visual C++ version 5.0 for the Windows NT version.

Note that Watcom 11.0b does not include, by default, DOS4GW Professional; it includes a lite version of DOS4GW that requires *dos4gw.exe* in the path when an application is compiled for that lite version.

To run the demo *play*; be sure that all default environment variables are set properly (see Environment variable section).

With *play* we can demonstrate multi-processing capabilities under Windows NT. The demo itself is not very useful, it just demonstrates that three different HOST processes can communicate simultaneously with the same Genesis. In demo *play* the display was set to `IM_DISP_MONO`. Here each HOST process displays their results in a different color plane. First process will display in `IM_DISP_BLUE`, second in `IM_DISP_GREEN`, and the third in `IM_DISP_RED`. The three processes should run at similar speed but it will depend mostly on the HOST task priority: a higher HOST task priority will send more commands than lower HOST task priority. At some point in the demo you may notice that some parts of the demo are skipped. This is because the 'C80 may not have enough memory to execute the commands sent to the board.

In the next lines we describe variables that need to be set to run the demo play properly. The *genesis.ini* file is normally already configured that way.

The demo uses a default DCF file set by:

```
DCFPATH=c:\genesis\dcf
DCFNAME=rs170.dcf           or
DCFNAME=ccir.dcf
```

A camera needs to be attached to the Genesis board (and must be compatible with the one defined in DCFNAME). The demo uses a default VCF file set by:

```
VCFPATH=c:\genesis\vcf
VCFNAME=vm105_60.vcf
```

The demo will load some TIFF files during execution from the path specified by TIFFPATH:

```
TIFFPATH= %GENESIS%\image
```

The demo will load a Shell:

```
COFFPATH=%GENESIS%\shell;c:\my_dir\shell;...
COFFLOADOPTION=                      (none required by
default)
COFFNAME=shell.out
```

To start the demo, type:

For DOS:

play<CR> (in directory %GENESIS%\demo\dos)

For Windows NT:

- Open a DOS command prompt. Set the background color of Windows desktop to 6 since the demo keys on color 6 of the Genesis overlay.
- Be sure that the DOS command prompts have the Genesis environment variables properly set.
- go in directory %GENESIS%\demo\winnt

For single-processing:

- play.exe<CR>

or for multi-processing:

- playc.bat<CR> : this batch file will start three instances of play.exe but with a different parameter. First one is started with COLOR option to force display in Color mode. The two others are started with GREEN and RED, respectively. When the COLOR/Blue thread is started, go back to the parent DOS Commands prompt to start the other two processes. At this point you should see three different processes running.

Next follow the instructions while the demo is running.

More demos are available in directory %GENESIS%\examples\host\misc for the source. Directory %GENESIS%\examples\host\dos contains executables

for DOS. Directory %GENESIS%\examples\host\winnt contains executables for Windows NT.

For DTK users, examples can be found in %GENESIS%\examples\host\thresh and %GENESIS%\examples\local\thresh. Both HOST and LOCAL have to be compiled and linked to run it. Follow the *readme* file in the respective directory to compile and link this example.

6. Appendix

6.1 IMCC.BAT reference

Synopsis:

Compile source file(s) (.C, .CPP, .ASM, .PPC, or .S)
Compiling is compiler independent.

Usage:

```
imcc [/N] [/B] [option(s)] filename <CR>
imcc /c COMPILER_NAME<CR>
imcc * <CR>
imcc myfil*.c <CR>
imcc<CR>                                (Print this help file)
```

Options (order is irrelevant):

/N	nested make (internally used)
/B	force a build
/B-	do not force a build
/c COMP_NAME	Set compiler specific environment variables (use <i>imcc /sts</i> to see which ones have been added or changed). If no other option follows <i>imcc</i> , will only perform a compiler setup. All path settings required to compile a file is set by 'imcc'.
/d or /d-	Symbolic information or no symbolic information (default). Compile with full optimization when no symbolic information.
/ga	Generate assembly file from C file (output in LOG directory).

/np	Do not pause when error encountered during compilation.
/s SRCDIR	Where SRCDIR is the explicit path of the source directory.
/o OBJDIR	Where OBJDIR is the explicit path of the object directory. If /o is used and a library has to be created the /l option must be used.
/l LIBNAME	Where LIBNAME is the explicit library name to generate.
/r RSPFILE	Where RSPFILE is a special response file if default is not ok. Response files are used to configure Compiler switches.
/sts status	Display important environment variables as a
/rst	Force an update of the RAMDRIVE with the original 'imcc' tools.
/rev DIRNAME	Object revision sub-directory bellow OBJDIR.
/op LEVEL	Override default optimize level for some compiler.
/fd	Use RAMDISK as the temporary drive when compiling.

filename:

A filename with or without extension (in this case all .c, .cpp, .asm, .ppc and .s relatives to filename are compiled), or ALL (case irrelevant) to compile all source in the current directory or the directory specified by /s.

Wildcards (*, ?) are accepted in filenames; * can be used to regenerate everything.

Notes:

1-

By default 'imcc' assumes the following structure for the directories:

```

-LOCAL-- +BUFFER:           : source directory
          +-CDECODER        : source directory
          +-INIT            : source directory
          +-LOG             : logging dir. for BUFFER, CDECODER, ...
          +-MAKE            : make file dir. for BUFFER, CDECODER, ...
          +-OBJ----- +-BIG : object dir. for BUFFER, CDECODER, ...
          |               :           "
          +-OPTM-- +-BLOB   : source directory
          |               : source directory
          |               +-GAUGING
          |               +-OCR
          |               +-LOG
          |               +-MAKE
          |               : make file directory for BLOB, GAUGING, ...
...
          +-OBJ----- +-BIG : object directory for BLOB, GAUGING, ...
          |               +-LITTLE :           "

```

Directories' LOG, MAKE, and OBJ must be always there for the proper execution of 'imcc'. In directory OBJ, sub-directories are used to store objects for each compiler type. 'imcc' does not check for the existence of these directories; it presumes they already exist. If they do not exist an error will occur during 'imcc' execution. Use 'imcc /sts' and check the OBJDIR setting to find out the directory required for that compiler.

2-

A file named "objpath.rul" can be included during the execution of 'imcc'. If set, the OBJDIR line determines the path of the OBJECT and LIBRARY directories. Also, in that case, that line will override all previous settings of OBJDIR with the compiler default setting or by the /o option! Normally OBJDIR in objpath.rul is not used because 'imcc' sets' compiler specific values. If set, LIBNAME indicates the library to be created in the object directory. Also, in this case, that line will

override all previous settings of LIBNAME with the compiler default setting or by the /l option!

Some objpath.rul examples:

```
-----  
| # OBJPATH.RUL (empty file, default object path used, no library)  
-----
```

```
-----  
| # OBJPATH.RUL (used that path for objects, no library)  
| OBJDIR=..\obj\dos  
-----
```

```
-----  
| # OBJPATH.RUL (used that path for objects, no library)  
| # $(OBJDIR) is replaced with the default compiler specific path  
| OBJDIR=.$(OBJDIR)  
-----
```

```
-----  
| # OBJPATH.RUL (default object path used, create library mylib.lib)  
| # library is put in default OBJDIR directory  
| LIBNAME=mylib.lib  
-----
```

```
-----  
| # OBJPATH.RUL (default object path used, create library mylib.lib)  
| # library is put in that absolute path  
| LIBNAME=d:\genesis\lib\obj\doswat\mylib.lib  
-----
```

```
-----  
| # OBJPATH.RUL (used that path for objects, create library mylib.lib)
```

```
| OBJDIR=..\obj\dos  
| LIBNAME=mylib.lib  
-----
```

3-

A file named "libname.rul", if found in the OBJDIR directory, is used to explicitly define the library associated with that directory. This file contains the definition of the library to create. The Syntax of this file is identical to the "objpath.rul" file presented above.

A libname.rul example:

```
-----  
| # LIBNAME.RUL (create library mylib.lib)  
| LIBNAME=mylib.lib  
-----
```

4-

A group of source directories can be compiled by creating a makefile in the root directory below the make directory of this group of source directories.

For example:

```
-----  
| # Makefile for a nested make  
| # created by cmkfile.bat  
|  
| DIR2DO = HL LL  
|  
| .PRECIOUS : $(DIR2DO)  
|  
| all : $(DIR2DO)  
|
```



```

| HL :
|   @cd .\${@
|   $(MAKE) /f .\make\makefile
|   @cd ..\
|
| LL :
|   @cd .\${@
|   $(MAKE) /f .\make\makefile
|   @cd ..\

```

Then, to compile all sub-directories just type at the root directory:
 imcc [option(s)] *
 (options will be used for all sources in this case).

To simplify the creation of these makefiles, a batch file, *cmkfile.bat*, is available in your RAMDRIVE. To use this batch file simply run *cmkfile*<CR> from the proper place. *cmkfile* without parameters creates a make file only for the current level. When you wish to create all make file in sub-directories simply type *cmkfile ALL*<CR>.

5-

Two environment variables have to be set before calling imcc.bat:

```

set BATCH=d:\genesis\util\    Where to find batch and batch related files
set RAMDRIVE=v:\              Temp drive (use Virtual drive for speed)

```

If not set the above defaults will be used (NOTE TRAILING " \ ")

If a Virtual Drive or RAMDISK is not available, a real drive is required. Because some commands tend to be long (close to 128 characters) do not set RAMDRIVE to something like c:\tools\imcc\ . This long RAMDRIVE definition will confuse some non-NT executable. Instead do a substitute to c:\tools\imcc\ and set RAMDRIVE to that substituted drive. Example:

```
subst e: c:\tools\imcc
set RAMDRIVE=e:\
```

6-

Five environment variables can be set before calling *imcc.bat*. These variables will force a particular method of compilation instead of using options.

DEBUG	nothing or /d- for no symbolic information, /d for symbolic debugging (same as option /d)..
SRCDIR	The directory of the source (as /s)
OBJDIR	The directory of the object and library (as /o). If used the file "objpath.rul" will not be.
LIBNAME	The library name (as /l)
RSPFILE	The response file (as /r)

Options, if used, will override these environments variable during *imcc.bat*.

7-

Files created in RAMDRIVE during the batch file processing:

rule.mak	Contains the multiple options and a response file
allfile.src	Description of file(s) to construct
tmp.rsp	A merged response file.
Logname	Used when error occurred in make

8-

Default files used during batch/makefile processing:

make\makefile	A makefile found in the make directory if defined to build a group of directories.
%RAMDRIVE%\makefile	A makefile that controls all the make process
objpath.rul	Defines the rules for the object and library destination

compiler.typ	A list of supported compilers with specific macros for each compiler or assembler.
--------------	--

9-

Each compiler has a pair of files .RSP and .SET. .RSP file is a response file that configures the switches of the compiler (common ones to unify compilation). .SET file is used to set environment variables required by the compiler. Also, for the compiler/linker, a file with extension .FMT defines the format of the link list.

10-

For efficiency some tools are transferred to the RAMDRIVE:

beep	Generate a programmable sound on machine speaker
bset	Tool to print and set environment variables (DOS & WIN3.11)
chkerr	Check error for some compilers that don't return an errorlevel
dos2unix	Convert a DOS file to a UNIX file.
Errimcc	Force errorlevel to a certain value
gawk	Tool to do pattern scanning and processing language
ifexist	Check if file or directory exists (replace buggy 'if exist' in DOS, Win 3.1, and Win NT)
nmake	Microsoft NMAKE.
Readme	Text browser

11-

Restriction:

In the .SET file, BATCH must end with a slash.

In the .SET file, all paths where we wish to check dependencies have to be specified in lower case and the rest in UPPER CASE for INCLUDE and C_DIR environment variables.

RAMDRIVE (probably in your autoexec.bat) must end with a slash.

12-

QUICK START:

- a- Create a RAMDRIVE on your system. In config.sys
device=c:\dos\ramdrive.sys 1024 /E
This RAMDRIVE should be drive D:
Set RAMDRIVE somewhere appropriate. Example:
'set RAMDRIVE=D:\' in your autoexec.bat.
Before you install the RAMDRIVE be sure that no drive is
assigned as a shared drive in Windows For Workgroup (if so the
RAMDRIVE defined in the config.sys will conflict with that
shared drive in a DOS Box).
- b- Set BATCH to the path of 'imcc' tools. Example
'set BATCH=d:\GENESIS\UTIL\.'
- c- Add to your path the path of 'imcc' tools.
Example
PATH=%PATH%;%BATCH%;
- d- Call 'imcc /c COMPILER_NAME' to switch to that compiler.
- e- Call 'imcc FILE_NAME' to compile FILE_NAME for the
compiler chosen at step 4.
- f- You may wish to always preset 'imcc' to a default compiler when
you login to the network. In that case place in a BATCH file run by
the LOGIN procedure a call to 'imcc'. Example:
call imcc /c lmvp
This will preset 'imcc' for the little endian MVP compiler.

13-

QUICK COMPILER SETTING:

Do `imcc /c COMPILER_NAME ...` to force a new compiler usage.

Example: `imcc /c phbc31<CR>` or
 `imcc /c phbc31 [other_options] your_file<CR>`

6.2 IMLK.BAT reference

Synopsis:

Link objects and libraries together. Link lists are compiler independent.

Usage:

```
imlk target [options_list_file] [options_list]<CR>
imlk<CR>                                (Print this help file)
```

Where options_list is any of the following Options and options_list_file is a file regrouping any of the following Options. Last option has precedence over previous ones.

Options:

/D YES NO	Symbolic debugging in target - YES is default
/OBJ objdir	Source of OBJs to link - ..\obj\comp_dir Default one is suggested
/L list	Link list (option may be used more than once) A link list, an object, or a library.
/REV release_directory	Use revision number in OBJs path - 1.0
/BIN LIB DLL file	Target type and name - [d:][\mypath\]myfile Default one is suggested
/DEF def_file	Definition file (used for some linkers)
/FMT fmt_file	OUTPUT format for that compiler -
myfmt.fmt	Default one is suggested
/RSP rsp_file	Response file to create for that linker - Default one is suggested
/DUP dup_file	Dump file for duplicate and wrong files -

/DRV drive	Default one is suggested Drive to use by default (d: is the default) -
/DOLNK linker	Default one is suggested Specific batch file called to link -
/TWOPASS YES NO	Default one is suggested Use YES to optimize code placement in memory; a two pass link is required. Use NO or don't use that switch to use default code location in memory.

Examples:

imlk myTarget /L mylst.lst /L common.lst

or with a response file myrsp.cmd containing lines:

```
/L mylst.lst  
/L common.lst
```

imlk myTarget myrsp.cmd

A response file may be combined with options, for example:

```
imlk myTarget myrsp.cmd /D NO
```

Objects or libraries can be passed to linker without declaring them in a link list file as in:

```
imlk myTarget /L myObj.obj /L common.lst
```

In that case myObj.obj can follow any rules defined in the following examples.

Link list format (examples):

```
.\file1.obj
```

in current directory

<code>\$(objdir)\file2.obj</code>	in drive:OBJDIR path from cur. dir.
<code>..\obj\\$(objdir)\file2_1.obj</code>	in drive:OBJDIR path from cur. dir.
<code>\src\\$(objdir)\file3.obj</code>	in drive:\src\OBJDIR path
<code>\src\\$(objdir)\\$(rev)\file4.obj</code>	in drive:\src\OBJDIR\REVISION

path

This is a comment line

<code>\src\obj\\$(objdir)\lib1.lib</code>	in drive:\src\OBJDIR path
<code>\src\obj\\$(objdir)\\$(rev)\lib2.lib</code>	in drive:\src\OBJDIR\REVISION

path

<code>\src\obj\little\file5.o</code>	same as file3 if OBJDIR matches
--------------------------------------	---------------------------------

'little'

<code>d:\src\obj\little\file6.o</code>	use that path as is lib3.lib in PATH
--	--------------------------------------

If OBJDIR=..\obj\big

<code>\$(objdir)</code>	is replaced by <code>..\obj\big</code>
<code>obj\\$(objdir)</code>	is replaced by <code>obj\big</code>

If a revision is specified like 1.0

<code>\$(objdir)\\$(rev)</code>	is replaced by <code>..\obj\big\1.0</code>
<code>obj\\$(objdir)\\$(rev)</code>	is replaced by <code>obj\big\1.0</code>

If no revision is specified

<code>\$(objdir)\\$(rev)</code>	is replaced by <code>..\obj\big</code>
<code>obj\\$(objdir)\\$(rev)</code>	is replaced by <code>obj\big</code>

Notes:

Always use \ in path (conversion is done for UNIX files)

Libraries and object files can be intermixed (reordering is done automatically)

Usage of \$(objdir)\\$(rev) permits automatic change of the link list according to the compiler in use and compiled revision.

If more than one link list is supplied by many /L options, they are merged into one link list. Precedence is from first to last one specified. If a file is encountered more than once, the first one is used.

As compiler specific object intermixing is not recommended (such as big\file1.obj with little\file2.obj), *imlk* considers the file that did not match OBJDIR as a wrong file if the drive is not explicitly specified (as for file5 in examples above).

Comments are permitted on lines containing no object or library. A comment line starts with #. See the link list format above for example.

6.3 Default *genesis.ini* file

```
#=====
#
# Genesis.ini
#
# Notes:
# 1- When IgnoreEnvVar is set to 'no' SW will first search in environment
#    variables and if not found, in genesis.ini file.
#    When IgnoreEnvVar is set to 'yes' SW will first search in genesis.ini
#    file and if not found or commented, in environment variables.
#
# 2- Any line, except section line ([...]), can be commented with #.
#    Default value for a line that has a string parameter is a NULL pointer.
#    Default value for a line with a number as parameter value is 0.
#
# 3- GenLib will search for this file according to the following algorithm:
#    First in path specified by environment variable GENINIPATH,
#    Second in path specified by environment variable GENESIS under
#        directory 'shell' (as in c:\genesis\shell when
#        GENESIS=c:\genesis),
#    Third in the current directory,
#    Fourth in path specified by environment variable COFFPATH,
#    Fifth in path specified by environment variable PATH.
#
#    If file genesis.ini can't be found then GenLib will not be able to
#    establish communication with genesis board(s) since vital information
#    found in section 'PrivateGlobalSetting' can't be read.
#
# 4- ApplicationMessaging is to control GenLib internal print. Under NT all
#    internal prints are piped to GenCout. Under DOS there is no piping. Valid
#    values are:
#    _IM_NO_PIPE      No printout.
#    _IM_DEFAULT_PIPE Print in Pipe (according to GenCout under NT).
#    _IM_KEEP_PIPE    Under NT keep GenCout open for debugging purpose
#                    (or according to GenCout setting)
#
# 5- GenLib will always update 'genesis.ini' when Hardware profile changes. If
#    for any reason you need to maintain what is currently defined then set
#    AutoRescanIni to NO.
```

```
# If you add or remove a back plane or cable from the VM, the GRAB, or the
# Raster Blaster port, the SW will not detect these changes and may report
# wrong information.
#
# 6- VM interface can work at 25 MHz with a flat cable or at 33 MHz with the
# VM backplane. However in may be required in some circumstances to override
# the maximum speed of the VM interface. This feature is mostly
# useful when the long VM backplane is used. 25 MHz is always selected for a
# flat cable. Speed selection is done at the first imDevAlloc after a power
# ON / or system reset. If you change this setting you will have to reboot
# your system to make it active.
#
# 7- Display information on the board that is currently downloaded: board
# interconnections, board revision, board modification, manufacturing info,
# etc. Note that some variations of the Genesis family may not provide
# that information.
#
# 8- When debugging your application it may be important to stop when a
# synchronous function doesn't end normally. In that case set that field to
# YES. In production mode, looping for user intervention
# to end the execution of an endless operations, is not really usefull. In
# that case set that field to NO. The synchronous operation will end and
# will log an Application error that can be retrieved with
# imAppCatchError().
#
#
# Revision history:
#
# January 8 1997 14:00:00
# First release
#
# April 4 1997 10:24:00
# Release 1.1
#
# November 18 1997 9:56:00
# Release 1.18
#
# December 3 1997 10:46:00
# Release 1.2
#
# May 1998 12:50:00
# Release 1.25
#
```

```

[GlobalSetting]
IgnoreEnvVar=yes                # See Note 1
ApplicationMessaging=_IM_DEFAULT_PIPE # See Note 4
AutoRescanIni=yes              # See Note 5
MaxVmDevicesAt33=7
VmSpeedLimit=33                # See Note 6
DisplayBoardInfo=no           # See Note 7
CommunicationDebugMode=no      # See Note 8
LargestDmaBlock=0x04000000
DisableDefaultMappingOfVirtualBuffer=no
ReservedHostSpaceForLcBuffers=0x10000
SizeOfTmpBufferForHostGrab=0x4000
InterruptDrivenOSB=no
TimeOutCommunicationGet=15      # 15 seconds

```

```

[EnvSetting]
CoffPath=c:\genesis\shell
CoffName=shell.out
CoffLoadOption=F+
FpgaPath=c:\genesis\shell
TifPath=c:\genesis\image
JpgPath=c:\genesis\image
MimPath=c:\genesis\image
DcfPath=c:\genesis\dcf
DcfName=rs170.dcf
VcfPath=c:\genesis\vcf
VcfName=vm107_60.vcf
DiagDumpFile=c:\genesis\shell\diagdump.gsp

```

```

[PrivateGlobalSetting]
# WARNING: These fields are used to configure Genesis Board. Do not remove or
# change any fields without Matrox advise.
# When these fields are not preceded by primary. or display., then they apply
# for both VIA.
# When preceded by ppb. they apply to the PCI-TO-PCI Bridge
ppb.pcstatuscmd.serren=0
ppb.pcstatuscmd.perrsp=0
ppb.pcstatuscmd.bsmstr=1
ppb.pcstatuscmd.memspc=1
ppb.pcstatuscmd.dpardctd=1
ppb.pcstatuscmd.rtrgtab=1
ppb.pcstatuscmd.rmstrab=1

```

ppb.pcstatuscmd.ssyserr=1
ppb.pcstatuscmd.dctdperr=1
ppb.pcpbctrl.pwipostm=1
ppb.pcpbctrl.pwpostm=1
ppb.pcpbctrl.prmpprefm=0
ppb.pcpbctrl.prpprefm=3
ppb.pcpbctrl.swipostm=1
ppb.pcpbctrl.swpostm=1
ppb.pcpbctrl.srmpprefm=0
ppb.pcpbctrl.srpprefm=3
ppb.pchdtmr.pchlnsz=4
ppb.pchdtmr.pclattmr=128
ppb.pcpbretry.retrycross=0
pcstatuscmd.serren=0
pcstatuscmd.perrsp=0
pcstatuscmd.bsmstr=1
pcstatuscmd.memspc=1
pcstatuscmd.dpardctd=1
pcstatuscmd.rtrgtab=1
pcstatuscmd.rmstrab=1
pcstatuscmd.ssyserr=1
pcstatuscmd.dctdperr=1
pchdtmr.pclattmr=247
hbusctrl.intseen=0
hbusctrl.intpeen=0
hbusctrl.pe2seen=0
hbusctrl.viarcmd=2
hbusctrl.mvprcmd=0
memctrl.refrate=47
macctmng.disctmr=10
primary.macctmng.mrasmax=31
display.macctmng.mrasmax=256
macctmng.mmingnt=128
display.dclksel.dclksel=150
display.dhdelay.dhdelay=4
hwftrsh.hwftrsh=8
hrftrsh.hrfwtrsh=7
vrftrsh.vrfwtrsh=7
vrftrsh.vrftrsh=2
primary.hrftrsh.hrftrsh=8
display.hrftrsh.hrftrsh=10
primary.mrftrsh.mrfwtrsh=7
primary.mrftrsh.mrftrsh=4

```
!if 430FX
@echo. 430FX Host bridge
!endif
```

```
!if 430HX
@echo. 430HX Host bridge
!endif
```

```
!if 430TX
@echo. 430TX Host bridge
!endif
```

```
!if 430VX
@echo. 430VX Host bridge
!endif
```

```
!if 440FX
@echo. 440FX Host bridge
!endif
```

```
!if 440LX
@echo. 440LX Host bridge
!endif
```

```
!if 450KX
@echo. 450KX Host bridge
!endif
```

```
#
=====
# Advance feature:
# -----
#
# With the Ini file it is possible to program or read any PCI configuration
# space of any PCI device.
#
# In the following syntax:
# pci.          is mandatory at the beginning of the line
# DevId/VendId  is the Device + the Vendor Id of the PCI device to
#               access.
# RegOffset     is the register offset in bytes in the PCI
#               configuration space.
```

```

# BitMask      is a group of successive bits that represent the Bit
#              mask of the field to access.
# FieldValue   is the value to be written at the BitMask.
# RegValue     is the Double Word value to be written at the RegOffset.
#
# If line is equal to:
# 1- pci.DevId/VendId.RegOffset.BitMask=FieldValue or
#    pci.0x122d8086.0x40.0x00007000=0x3
#    A bit field access is executed to that register. Value is
#    written in field marked by the BitMask. Register offset is
#    rounded to Double Word offset.
#
# 2- pci.DevId/VendId.RegOffset=RegValue or
#    pci.0x122d8086.0x44=0x12345678
#    Value is written in the Double Word pointed to by the register
#    offset. Register offset is rounded to Double Word offset.
#
# 3- pci.DevId/VendId.RegOffset or
#    pci.0x122d8086.0x44
#    Register is read and printed at the console.
#
# 4- pci.DevId/VendId or
#    pci.0x122d8086
#    If device is found it is printed at the console.
#
# If device doesn't exist the command is skipped.
# All devices that respond to the same Device and Vendor Id are
# written or read the same way.
#
# WARNING:
# Use this feature with extreme caution. Never use this feature without
# Matrox advise.
#
#

```

```

=====

pci.0x00221014.0x18.0xff000000=0xc0
pci.0x00221011.0x18.0xff000000=0xc0
pci.0x00241011.0x18.0xff000000=0xc0
pci.0x00221011.0x0c.0x0000ff00=0xc0
pci.0x00241011.0x0c.0x0000ff00=0xc0

pci.0x00221011.0x40.0x000000ff=0x10

```

pci.0x00241011.0x40.0x000000ff=0x10

```
#
=====
# Comment on Node(s) section(s):
# -----
#
# Values defined in 'GlobalSetting', 'EnvSetting', and 'PrivateGlobalSetting'
# apply for all nodes unless modified in each specific node.
#
# Add for each node any fields of 'EnvSetting' or 'PrivateGlobalSetting' to
# configure that specific node.
# Prefix primary. or display. can be added to any element of
# 'PrivateGlobalSetting' added here. Each VIA can be configured individually.
# When these fields are not preceded by primary. or display., then they apply
# for both VIA.
# When preceded by ppb. they apply to the PCI-to-PCI Bridge
#
# WARNING:
# Node(s) section(s) is(are) configured dynamically first by imDevAlloc().
# Any field of 'EnvSetting' or 'PrivateGlobalSetting' set in 'Node' section
# is lost after Node section(s) is(are) updated.
# Only 'Node' section is affected by this, other sections are preserved.
#
# DisplayMode can have the following value:
# IM_SINGLE_SCREEN, IM_DUAL_SCREEN, IM_DUAL_HEAD, IM_NONE
#
# VmBusSpeed can be set to:
# 25 or 33
#
# GrabPortConnection, VmPortConnection, and RasterBlasterConnection is a list
# of DeviceNbr attached together in the form:
# 0x001004,0x000d04,...
# No connection when the field is set to 0.
#
# If an external VM device is added in the VM chain set in VmExternalDeviceId
# the Device Id associated to that device so GenLib software will be able to
# chain that Device with Genesis device(s). More than one device can be added
# in the chain. For examples:
# 15 (for 1 external device with Device Id of 15) or
# 15,14,13 (for 3 external devices).
# No external device when the field is set to 0.
```



```
# Node in the same System must have identical VmExternalDeviceId information.  
#  
# After NodeId field there are two fields, Section and Node that are the System  
# and Node parameters of imDevAlloc(). Only the NodeId is relevant here. System  
# and Node printed in 'Node' section is there just as an explanation of the  
# NodeId. The NodeId is determined according to the following rule:  
#   NodeId = (((System + 1) << 4) + (Node + 1))  
# If for any reason the NodeId needs to be changed there is no need to change  
# the System and/or Node lines since Software ignores them.  
#  
#
```

```
=====
```