



GURU

Release Notes

October 1997

Version 7.0

Legal Notices

Copyright © 2004, 2005 Savitar Corporation.

These manuals are provided for the use of customers and customer employees. The entire contents have been copyrighted by Savitar Corporation. Copying, duplicating, selling, or other distribution of the software is a violation of the copyright laws. Reproduction by any means of the software or documentation is prohibited except as permitted in a written agreement with Savitar Corporation.

TITANIUM, GURU, and all other Savitar products including legacy **mdbs** products are copyrighted computer software, and all rights, including intellectual property rights, are reserved by Savitar Corporation. Use of software requires an End User License. Please refer to the End User License Agreement for the corresponding terms and conditions.

TITANIUM is protected by U.S. Patent Nos. 5,611,076 and 5,713,014 in addition to foreign patents and applications pending.

Limitation on Warranties and Liability

The documentation in these manuals is provided for reference purposes only. Savitar Corporation has taken great pains to ensure that the manuals reasonably conform to the software. However, Savitar Corporation reserves the right to make changes in specifications and other information in these publications without prior notice.

Savitar Corporation makes no warranties, either express or implied, respecting the software and documentation or its quality, performance, merchantability, or fitness for any particular purpose. Savitar Corporation software and documentation is licensed for use "as is."

Savitar Corporation shall not be liable to the user or any other person or entity with respect to any liability, loss or damage caused by, or alleged to be caused by, the software and documentation or its use, whether direct, incidental, or consequential, according to the terms of the license agreement.

Trademarks

TITANIUM[®] and GURU[®] are a registered trademarks of Micro Data Base Systems. MDBS IV[®], MDBS III[®], KnowledgeMan[®] and Object/1[®] are also registered trademarks of Micro Data Base Systems.

Other brands and product names may be trademarks or registered trademarks of their respective holders.



Contents

Chapter 1

INTRODUCING GURU 7.0

System Requirements	1:2
RAM	1:2
Disk	1:2
Visual C++	1:2
Technical Notes	1:3
Edit/Copy/Paste functionality	1:3
CLINK Examples	1:3
New TBL routine	1:3
Year 2000 Features	1:5
Year 2000 Compatibility for Two-Digit Dates – E.Y2K2	1:5
Year 2000 Compatibility for Four-Digit Dates – E.Y2K4 ...	1:6



Contents

Chapter 1

INTRODUCING GURU 7.0

GURU 7.0 is a 32-bit version of GURU that runs as a native Win32 console application on Microsoft Windows 95 and Windows NT. GURU 7.0 is fully compatible with previous versions of GURU such as GURU 6.0 DPMI and GURU 3.1 MSDOS. GURU 7.0 will run most previously existing KnowledgeMan and GURU application code as is, with no changes. This release of GURU 7.0 includes the following new features:

- ◆ 32-bit versions of G3C, GLIB, and other utilities
- ◆ A new graphical ruleset editor (GRSE). See chapter 3 of “Building Expert Systems with GURU for more information.
- ◆ An enhanced KEYMAP() function provides unique keycodes for key combinations that were overloaded under MSDOS. Backward-compatibility is maintained with almost all MSDOS extended keycodes (using lead-in code of \255), except for those keys that generate system interrupts such as Ctrl-C, Ctrl-Break, and Alt-Print Screen.
- ◆ Features to address Year 2000 date issues in existing KnowledgeMan and GURU programs. See the **Year 2000** section of this document for more information.
- ◆ A new graphical installation
- ◆ A new TBL.DLL low-level database driver with increased performance over the TBL library embedded in 16-bit KnowledgeMan and GURU, especially with disk-intensive operations such as CONVERT. A 32-bit TBL.LIB library for accessing .ITB files from C programs is also included with this release; this library is an import library for TBL.DLL.
- ◆ A Win32 CLINK library is included for extending GURU functionality with Microsoft Visual C/C++ or any other language that can create Win32 dynamic link libraries (DLLs). Several CLINK examples are included with GURU 7.0 that show how to integrate GURU with the Win32 SDK and take advantage of graphical interfaces and other Windows capabilities. See CLINK Examples, in the Technical Notes section, below.
- ◆ The GRAPH and COMM capabilities described in the documentation are not currently supported in GURU 7.

System Requirements

System Requirements

RAM

GURU in this environment requires an approximate minimum of 2 megabytes of RAM in order to load. The amount of memory required to run a particular GURU application varies depending on the application's use of memory. **mdbs** recommends the minimum RAM needed to run the operating system with good performance characteristics: 16 megabytes for Windows 95 or 32 megabytes for Windows NT.

For systems with many applications running simultaneously, increase the amount of RAM to 32 megabytes for Windows 95 or 64 megabytes for Windows NT.

Disk

A full installation of GURU 7 requires 8 megabytes of disk space. A typical installation requires 6 megabytes, and a minimum installation requires only 4 megabytes

An IBM-compatible PC is required, with an Intel 486 /Pentium compatible CPU running Windows 95 or Windows NT 4.0 (or greater)

Visual C++

This release requires Microsoft Visual C++ 5.0 for CLINK and TBL

Technical Notes

Edit/Copy/Paste functionality

To enable the Edit/Copy/Paste functionality within the GURU window, disable the Fast Pasting option for the MS-DOS prompt command window. If GURU is already started left click on the system, menu (small MS-DOS icon next to the window title bar), choose Properties, then choose the Misc tab. Click to remove the check mark in the Fast Pasting check box in the “Other” panel.

CLINK Examples

Two CLINK example programs are provided to demonstrate how to call Win32 APIs. Any Win32 API may be used from GURU through the CLINK interface, thus enabling GURU to make use of any Windows interface or facility. Compile and linked DLLs of these examples are provided for use with GURU even if no C/C++ compiler is available. Copy the DLL to the GURU directory before loading within GURU code.

The first CLINK example, OFILEDLG.DLL, is used in the GURU Commander to call the Windows Open Files dialog. Source code for this DLL is included in the CLINK subdirectory. Use this as an example of how to call Win32 APIs to extend GURU’s character-mode interface with graphical user interface elements. See the CM_FILE.KGL Commander source file in the GUIDE\SRC directory for usage of the OFD_BOX, OFD_NUM, OFD_GET, and OFD_CLEA function calls contained in the OFILEDLG library.

The second CLINK example, MEM32.DLL shows how to get information about system resource usage. The files MEM32.DLL, MEM32.IPF, and MEM32.C files in the CLINK subdirectory show how this function is implemented and called.

New TBL routine

A new TBL routine, tbl_version, has been added to GURU 7.0. Details of the routine are provided on the following page.

tbl_version

Name **tbl_version()** – returns the TBL version number

Synopsis:

```
#include      "tbl.h"
char* tbl_version();
```

Return Values:

returns a string representing the current TBL version number

Arguments:

none

Description:

This function returns the version number of TBL. Check this number to make sure that it is at least as high as the version of TBL that the application has been tested with. Older versions of TBL may not work or may corrupt data.

Example:

```
#include <string.h>
#include <tbl.h>
void version_check()
{
    char * tbl_version_str;
    /* tbl version string has format */
    /* of major.minor.internal.other */
    /* for example, 700.000.015.000 */
    tbl_version_str = tbl_version();
    printf("TBL Version = %s\n", tbl_version_str);
}
```

Year 2000 Features

The upcoming century change has a major impact on computer programs that process calendar dates, as most programs of the past 50 years have implemented calendar years as 2–digit quantities ('96, '97, and so on) with an implied year in the 20th century (19xx).

KnowledgeMan and GURU applications are no exception to this. While the KnowledgeMan and GURU tools have always processed dates internally as 4–digit quantities, many KnowledgeMan and GURU application programs have been implemented with 2–digit dates to save memory, database space, and processing time.

Year 2000 compatibility in GURU 7.0 has been designed with two goals in mind:

- ◆ Provide a compatibility mechanism for applications using 2–digit years to allow these applications to run correctly on and after the Year 2000 without recoding.
- ◆ Provide full 4–digit year processing for applications that require it, allowing these applications to be recoded for 4–digit years.

To meet these two goals, two environment variables have been added to GURU 7.0. The first environment variable, E.Y2K2, addresses the first goal by controlling the interpretation of 2–digit dates. The second environment variable, E.Y2K4, addresses the second goal by controlling the generation of dates with 4–digit years.

Year 2000 Compatibility for Two–Digit Dates – E.Y2K2

When interpreting a 2–digit date such as “10/10/95”, a computer program can interpret the year '95 as either being in the 20th century (1995 A.D.), the 21st century (2095 A.D.), or some other century. The TOJUL() function in KnowledgeMan and GURU has until now interpreted such dates as being in the 20th century (19xx). With GURU 7.0 the environment variable E.Y2K2 controls how the TOJUL() function interprets 2–digit dates.

By default the value of E.Y2K2 is 50. With E.Y2K2=50, dates in the range 50–99 are interpreted as 20th century (19xx), and dates in the range 00–49 are interpreted as 21st century (20xx). For example, TOJUL(“10/10/95”) would return a Julian date corresponding to November 10, 1995, while TOJUL(“10/10/01”) would return a Julian date corresponding to November 10, 2001.

With this default, most existing KnowledgeMan and GURU applications can continue to run without changes through the year 2049. The default can be adjusted for specific applications. For example, consider a database with historical data or birth dates having 1938 as the year of the earliest stored date.

Year 2000

This application could run up to the year 2037 without further code changes by setting E.Y2K2 to 37. Then dates in the range 00–37 would be interpreted as 21st century (2037), while dates in the range 38–99 would be interpreted as 20th century (1938).

If the completely backward-compatible behavior of interpreting all 2-digit dates as 20th century (19xx) is required, that can be accomplished by setting E.Y2K2=0.

Year 2000 Compatibility for Four-Digit Dates – E.Y2K4

Some organizations require that Year 2000 changes include recoding all dates as 4-digit years. This requires application code changes. For these needs, the environment variable E.Y2K4 can be set to TRUE (the default is FALSE). With E.Y2K4=TRUE, the function TODAY() returns a date string containing a 4-digit year rather than a 2-digit year. The #DATE utility variable contains a date string with a 4-digit year as well.

Corresponding code changes in the application may be required to handle 4-digit years. Screens may need to be adjusted for a date string that is longer now by 2 characters. Calls to the TODATE() function to generate date strings must be adjusted so that TODATE() generates 4-digit years (by changing the 3rd argument of TODATE() to a 4, as in TODATE(juldate, “/”, 4)). Code that interprets or manipulates date characters through string operations such as SUBSTR() must be changed appropriately.