

Update notes

Last updated 8-2-2010

Windows 98/ME/NT/2k/XP/Vista/Win7 version

See the new links at our website for the most recent downloadable device driver, DLL and support files, documentations, and simplified sample applications.

NOTE: As of 2-18-10 this documentation includes both the Win98/ME as well as the Win2k/XP/Vista/Win7 update information.

NOTE: Although the DLL is same for 64-bit and 32-bit, the driver is different. The driver is signed for 64-bit systems while the 32-bit is not and is not supported on 64-bit operating systems such as 64-bit Vista or 64-bit Win7 running in 64-bit mode.

=====
Update information (oldest at bottom)
=====

8-2-10

1.) Fixed problem with sending of rate. Under certain circumstances, during low-level driver activity beyond our control, the sending of the rate could fail. This problem has been fixed in this version of the DLL.

2.) Version information:

VB App:	2,4,2008
VC Console App:	8,30,2002
VerChk utility:	12,14,2004
DLL (Win2k/XP/NT/Vista/Wn7):	8.2.2010.0
DLL (Win98/ME):	7,18,2008
64-bit DeviceDriver :	10,13,2008
32-bit DeviceDriver :	10,13,2008
MMKeep_LL.bin	4,14,2008
MMKeepDmp_LL.exe	7,12,2009
Hardware:	04

=====
Update information (oldest at bottom)
=====

NOTE that the 64-bit version compatible

DLL was not available prior to 2-18-10

2-18-10

1.) Improved debug output

2.) Version information:

VB App: 2,4,2008
VC Console App: 8,30,2002
VerChk utility: 12,14,2004
DLL (Win2k/XP/NT/Vista): 2.18.2010.0
DLL (Win98/ME): 7,18,2008
64-bit DeviceDriver : 10,13,2008
32-bit DeviceDriver : 10,13,2008
MMKeep_LL.bin 4,14,2008
MMKeepDmp_LL.exe 7,12,2009
Hardware: 04

2-9-10

1.) Fixed problem related to multi-threaded CPU's causing scan data corruption. Although the symptom often became evident after lengthy runs at high data rates, it could also happen just about anytime during a scan.

2.) Version information:

VB App: 2,4,2008
VC Console App: 8,30,2002
VerChk utility: 12,14,2004
DLL (Win2k/XP/NT/Vista): 2.9.2010.0
DLL (Win98/ME): 7,18,2008
DeviceDriver : 10,13,2008
MMKeep_LL.bin 4,14,2008
MMKeepDmp_LL.exe 7,12,2009
Hardware: 04

6-5-09

1.) Fixed variable settings used for call to EX_GetConfiguration(...). The variables returned by that call were not being correctly set by the various EX_ prefixed functions that make changes to the configuration of those variables.

2.) Version information:

VB App: 2,4,2008
VC Console App: 8,30,2002
VerChk utility: 12,14,2004
DLL (Win2k/XP/NT/Vista): 6.4.2009
DLL (Win98/ME): 7,18,2008
DeviceDriver : 10,13,2008
MMKeep_LL.bin 4,14,2008
MMKeepDmp_LL.exe 7,12,2009
Hardware: 04

=====

previous updates (prior to 2-9-2010)

=====

6-5-09

1.) Fixed variable settings used for call to EX_GetConfiguration(...). The variables returned by that call were not being correctly set by the various EX_ prefixed functions

that make changes to the configuration of those variables.

2.) Version information:

VB App: 2,4,2008
VC Console App: 8,30,2002
VerChk utility: 12,14,2004
DLL (Win2k/XP/NT/Vista): 6,4,2009
DLL (Win98/ME): 7,18,2008
DeviceDriver : 10,13,2008
MMKeep_LL.bin 4,14,2008
MMKeepDmp_LL.exe 7,12,2009
Hardware: 04

5-18-09

- 1.) Various modifications and fixes to the S1_ prefixed scan functions.
- 2.) Miscellaneous debug related modifications.
- 3.) Fixed a memory allocation error, that could cause a gpf under certain conditions, although there have not yet been any reports of this particular problem occurring as of this date in time.

4.) Version information:

VB App: 2,4,2008
VC Console App: 8,30,2002
VerChk utility: 12,14,2004
DLL (Win2k/XP/NT/Vista): 5,18,2009
DLL (Win98/ME): 7,18,2008
DeviceDriver : 10,13,2008
MMKeep_LL.bin 4,14,2008
MMKeepDmp_LL.exe 7,12,2009
Hardware: 04

4-17-09

1.) Various modifications to the S1_ type scan. The S1_ scan is for an easy interface to the 35b multiplexer. The main modification was the addition of a threaded interface. When we first implemented this scan, the calling app would have to wait within the DLL while data for all the channels was processed. With the threading feature, the app needs only to make a call into the DLL to begin the scan and then can call back in at any time to get data.

2.) The DLL function call, "GetDataLogOptions()" was listed in our API docs but was not exported from the DLL. That has been fixed.

3.) Made fixes to the handling of averaging set by call to EX_SetScanAvg(). There was some errors within the EX_ style scan thread if that function was called with an averaging value. The handling of the averaging by the scan thread is made available through that call as a convenience. However we recommend that you do your own averaging within your application.

4.) Fixed an error in SP_InitOneDev() that failed to check for a handle value of zero being passed to it. The application handle passed in that call is used for managing associated device ID's within the memory-mapped files and is also used for debug file related functionality. Another problem was fixed in that function - if a channel 7 was passed in the call, the channel was not being set from the default value of channel 0, to channel 7.

5.) Added a threaded device connect comprised of functions:

EX_ThreadedInitDev
EX_DLL_Init
EX_EndThreadedDevInit

This allows the application to make a single call in to begin the initialization of multiple devices and then another to check on the status of the initialization(s). See the documentation for those calls within the API documentation for the EX_ style functions at our website.

6.) Fixed a memory leak related to the application when debug file output was activated.

7.) Added additional content into the debug files. For example the value of ProcessPriorityBoost was not being recorded prior to this update. Also added content related to the PC hardware and OS. Also removed some less important or repetitive content.

8.) Improved DLL debug capability. We've improved the memory-mapped debug capability which we presented in the 7-21-08 update. We have also included the related files within the distribution disks, and downloads available at our website. To recap the explanation of the mem-mapped debug - when problems occur, the quickest way to get to the root of the problem is by setting "DebugLvl=2" in LL_USB.INI which is located within the Windows folder. Setting that entry to non-zero has always caused information that is valuable for troubleshooting to be written to a file within a folder that the DLL creates within the application folder called "M30x_DBG". Prior to the addition of the memory-mapped debug option, those files could become very large if a scan was performed with the debug level set to a value greater than 3, due to the disk activity required for the writing of the debug information. Now, more of the needed debug information is written to circular buffers within RAM memory, and then is written to the disk file when the application exits. There is still a small amount of debug information written to the disk during application runtime.

The use of the memory mapped debug can be further extended by including the files "MMKeep_LL.bin" and "MMKeepDmp_LL.exe" within the application folder. Those two files will allow the memory-mapped debug information to remain within memory even after a program crash and then that information can then be written to the disk the next time the application is loaded. With MMKeep_LL.bin in the application folder, MMKeepDmp_LL.exe can be executed at anytime while the application is running or even after the application has closed to display the debug information that has been written to the memory-mapped debug files. The output will appear within the application folder as a file with a name such as:

MMKeepDbg_Jul_21_07-54-46_2008.txt

showing the date and time that the file was written. In addition to that capability, certain scan related errors will cause the DLL to write such files automatically. If such an error occurs, the file written to the application folder, in addition to the date and time within the filename, will also have the clock time shown such as:

MMKeepDbg_Feb_25_09-45-28_2009_56640.txt

Note the 56640 within that name. That is the clock time of the error, and there could be several lines within the .txt file with that clock time beside them. Keep in mind that all of this functionality is only enabled when the "DebugLvl" entry within LL_USB.INI is non-zero.

9.) Version information:

VB App:	2,4,2008
VC Console App:	8,30,2002
VerChk utility:	12,14,2004
DLL (Win2k/XP/NT/Vista):	4,17,2009
DLL (Win98/ME):	7,18,2008
DeviceDriver :	10,13,2008
MMKeep_LL.bin	4,14,2008
MMKeepDmp_LL.exe	7,12,2009
Hardware:	04

=====
7-21-08
=====

1.) Fixed memory leak associated with the memory-mapped files used for storage of Device ID's and data.

2.) Added memory-mapped debug. Memory-mapped debug becomes active whenever the debug level in LL_USB.ini is set non-zero value. Suggested minimum debug level is 2. At that debug level the amount of debug always written to disk at that level will continue to be written, with additional debug being written to the memory-mapped files. The data written to the memory mapped files, will be written to the debug file when the application exits.

3.) Version information:
VB App: 2,4,2008
VC Console App: 8,30,2002
VerChk utility: 12,14,2004
DLL (Win2k/XP/NT): 7,18,2008
DLL (Win98/ME): 7,18,2008
DeviceDriver : 7,6,2004
Hardware: 04

=====
2-13-08
=====

1.) Added text on VB application main panel stating that rate can be set either by entering rate in drop down box, or selecting a rate from the drop down box.

2.) The handling of "StackedIrpCntOverride" by the DLL has been changed. If set to zero, one second's worth of IRPs will be stacked. Otherwise, this number represents 10ths of seconds of stacked IRPs. For example an entry of 30 yields 3 seconds worth of IRPs. IRP stands for "I/O Request Packet". The more IRPs that are "stacked" the more IRPs we request of the kernel with each request. In a way, this puts pressure on the kernel to process our requests as quickly as possible to prevent it from falling behind. As a result, this process provides our driver with more "power" over other drivers that may be hogging the system.

3.) Version information:
VB App: 2,4,2008
VC Console App: 8,30,2002
VerChk utility: 12,14,2004
DLL (Win2k/XP/NT): 2,6,2008
DeviceDriver : 7,6,2004
Hardware: 04

=====
10-3-07
=====

1.) Added "ToolTip messages" to many of the boxes and controls on the main application panel.

2.) Added an option to select plot screen foreground and background. There is a button at the bottom of the plot screen that can be pressed, causing a box to appear with color selection options.

3.) Fixed "tick" marks on plot screen. The vertical marks were being displayed incorrectly at very slow "seconds-per-screen" rates.

4.) Replaced reference to "DevRate" with text "Internal Data Rate". This has been a point of confusion by users. Remember that the Internal Data Rate is the rate at which the board is running and is only changed by either an initialization in which a new rate is passed to the board, or a specific call to change the rate. This is often confused with the scan rate. The scan rate is the time required to acquire all data points for one complete scan. Rather than being a configuration parameter, it is the result of configuration parameters. To calculate the scan rate, use the formula:
$$\text{data rate} / (\text{numScanChans} * 5)$$

5.) Version information:
VB App: 10,3,2007
VC Console App: 8,30,2002
VerChk utility: 12,14,2004
DLL (Win2k/XP/NT): 11,3,2006
DeviceDriver : 7,6,2004
Hardware: 04

=====
11-3-06
=====

1.) Repaired a bug that resulted from the added functionality within the DLL version "3,2,2006,0". The bug resulted in a

change to handling of the default stacked IRP's such that if the "StackedIrpCntOverride" within LL_USB.INI was left at zero, then the default stacked IRP count would be set to a default value of 1 rather than the previous setting of 20. This setting could result in scan halted errors as a result of microcode buffer wraps, causing the overflow of the microcode data buffer and a subsequent halting of the scan. There is a mechanism built into the DLL to halt the scan in the event of a microcode buffer wrap in order to prevent data corruption.

As is documented elsewhere, a microcode buffer wrap can occur if our device driver isn't serviced frequently enough to keep the data read from the device. Such a problem can occur if other drivers on the system do not release their hold on the system in a timely fashion to allow other drivers to get an adequate slice. Our use of stacking IRP's (I/O Request Packet) forces the kernel USB drivers to fetch the amount that we ask for with each request so even if another driver hogs the system, by the time we get our slice of time, the kernel has returned with a sufficient amount of data to compensate for the lost time away.

VB App: 3,14,2006
VC Console App: 8,30,2002
VerChk utility: 12,14,2004
DLL (Win2k/XP/NT): 11,3,2006
DeviceDriver : 7,6,2004
Hardware: 04

=====
3-22-06
=====

1.) Minor cleanup of DLL.

VB App: 3,14,2006
VC Console App: 8,30,2002
VerChk utility: 12,14,2004
DLL (Win2k/XP/NT): 3,22,2006
DeviceDriver : 7,6,2004
Hardware: 04

=====
3-16-06
=====

1.) The DLL has been updated with added functionality. Continue to visit our website for documentation concerning new API function calls:

<http://www.lawsonlabs.com>
and follow the links for your Model-30x device.

VB App: 3,14,2006
VC Console App: 8,30,2002
VerChk utility: 12,14,2004
DLL (Win2k/XP/NT): 3,2,2006
DeviceDriver : 7,6,2004
Hardware: 04

=====
10-20-05
=====

1.) Due to the size restrictions imposed by a floppy diskette, the most recent device driver, DLL (and support files), documentation, and sample applications have been placed at our website. We still include this floppy diskette with our distribution of hardware for those who may not have internet access. The files located on this diskette are the most recent excluding modifications to the DLL (and perhaps driver), required by the simplified sample applications and API added to the DLL to help simplify the tasks faced by the developer.

Please visit our website at:
<http://www.lawsonlabs.com>
and follow the links for your Model-30x device for more information.

VB App: 12,14,2004
LabView App: 4,10,2003
VC GUI App: 7,9,2003
VC Console App: 8,30,2002
VerChk utility: 12,14,2004
DLL (Win2k/XP/NT): 10,17,2005
DeviceDriver : 7,6,2004

=====
12-14-04
=====

- 1.) Miscellaneous fixes to Device Driver and DLL.
- 2.) Miscellaneous fixes to LabView and VisualC++ Console app
- 3.) Modified VB App so that it only requires the following entries in the LL_USB.cfg file for each device that you would like to run:
ID=555
RATE=100
CHAN=7
- 4.) Complete over-haul of Version Checker utility. Improved the utility with more trouble shooting capability to better handle larger diskdrives when doing it's searches. Also improved the interface so that now it has a few different options as to how much is displayed on the screen at any one time. Also added the writing of the information that it retrieves into a text file which can more easily read by the user, or sent to use for technical support.

VB App: 12,14,2004
LabView App: 4,10,2003
VC GUI App: 7,9,2003
VC Console App: 8,30,2002
VerChk utility: 12,14,2004
DLL: 12,2,2004
DeviceDriver : 7,6,2004
Hardware: 04

=====
12-16-03
=====

- 1.) Fix problem associated with users without specific rights being able to recover from device connect/disconnect in WinXP. Our previous versions used sections of the Registry restricted to certain levels of security so other users experienced access restrictions when our devices were either removed or added.
- 2.) Further improved handling of power issues - power loss, spikes, etc.
- 3.) Improved handling speed at which the software recovers from various errors conditions.
- 4.) Added the "Troubleshoot.htm" file to the documentation and updated the "IniFile.htm" file.

VB App: 12,9,2003
LabView App: 6,26,2002
VC GUI App: 7,9,2003
VC Console App: 6,19,2001
VerChk utility: 7,9,2003
DLL: 12,12,2003
DeviceDriver : 12,9,2003
Hardware: 04

=====
9-30-03
=====

- 1.) Modified HTML documentation in regard to reference to "Packet" meaning one grouping of data for all channels in scanning mode to "Scan". Did likewise in all "Class" related source code files.
- 2.) Many improvements to DLL function, DLL_InstanceInitDeviceList in regard to swapping of many boards to various USB cable connections. Prior to these fixes, some devices would become inaccessible after swapping and calling this function.

VB App: 8,15,2003
LabView App: 6,26,2002
VC GUI App: 7,9,2003

VC Console App: 6,19,2001
VerChk utility: 7,9,2003
DLL: 9,22,2003
DeviceDriver : 8,21,2003
Hardware: 04

=====
8-26-03
=====

1.) Improvements to Driver and DLL as well as minor changes to the VB sample application but only in regard to functions used by in-house testing.

VB App: 8,15,2003
LabView App: 6,26,2002
VC GUI App: 7,9,2003
VC Console App: 6,19,2001
VerChk utility: 7,9,2003
DLL: 8,21,2003
DeviceDriver : 8,21,2003
Hardware: 04

=====
7-9-03
=====

1.) Improvements to Driver and DLL as well as minor changes to the VB and VC-GUI sample applications. Changes to Driver were to bring it closer to passing the strict WinXP compatibility tests. Some of the changes included handling of hibernation and sleep as well as some changes to our .inf file which caused us to now have a separate one to be used for WinXP/Win2k/WinNT which is not the same as the one now used for Win9x/ME.

2.) Modified our VerChk.exe utility to work with a new driver security feature that had to be added for the WinXP compatibility tests and because of the new driver feature previous versions of the VerChk.exe utility will not work with newer versions of the DLL and Driver.

3.) Added new entry to LL_USB.INI to allow DLL (and it's calling application) to acquire a higher system timeslice resolution. The entry is called, "ProcessPriorityBoost", and is documented both within the LL_USB.INI file as well as the "IniFile.htm" file which is part of the documentation package .

VB App: 6,10,2003
LabView App: 6,26,2002
VC GUI App: 7,9,2003
VC Console App: 6,19,2001
VerChk utility: 7,9,2003
DLL: 7,9,2003
DeviceDriver : 7,9,2003
Hardware: 04

=====
4-7-03
=====

1.) Lots of improvements to Driver and DLL as well as minor changes to the VB sample application.

2.) Modified our VerChk.exe utility which had an error in the checking of older versions of our DLL which weren't compatible with the capabilities of the VerChk utility, as well as an error in the listing of all copies of the the DLL and driver currently on a system.

3.) Added new entry to LL_USB.INI to allow DLL (and it's calling application) to acquire a higher system timeslice resolution. The entry is called, "ProcessPriorityBoost", and is documented both within the LL_USB.INI file as well as the "IniFile.htm" file which is part of the documentation package .

VB App: 3,24,2003
LabView App: 6,26,2002
VC GUI App: 5.21.2002
VC Console App: 6,19,2001
VerChk utility: 3,19,2003
DLL: 4,2,2003
DeviceDriver : 4,2,2003

=====

11-6-2002

=====

1.) Repaired problem in driver related to swapping two different device IDs using the same cable connection to the PC when there is only a single device connected at any one time. For example, if you had two of our devices sitting in front of you and had connected a cable to one and initialized it, then disconnected the cable from it and connected it to the other device and attempted to initialize it, it would likely fail to initialize until you disconnected all devices and exited the application which would unload both the DLL and Device Driver.

2.) Added a separate DLL function call to improve writing to the device while in scanning mode. At the time of this writing it's usage is recommended for sending digital output while scanning. Using the new function greatly reduces processor time slicing and memory efficiency as well as speed.

3.) Due to improvement to the driver we now suggest starting with a setting of only 2 or 3 for the StackedIrpCntOverride entry in the LL_USB.ini file (included in the installation) if you experience "microcode buffer wrap" issues associated with other drivers that occasionally "hog the system" such as "pinball" in WinME and later versions of Windows. We had previously suggested a much higher setting. Keep in mind that the recommended setting for ScanSystemBoostLevel in the same .ini file is still 20 if there is any problems such as the "microcode buffer wrap" issue. Setting that particular entry to a high value has no adverse effect on the functionality of the system so it's fine to just always keep it at that value.

VB App: 10,14,2002
LabView App: 6,26,2002
 VC GUI App: 5.21.2002
 VC Console App: 6,19,2001
 VerChk utility: 5,9,2002
DLL: 11,6,2002
DeviceDriver : 11,6,2002
 Hardware: 04

=====

6-26-2002

=====

1.) Improved handling of setting of "scan HALTED" flag in DLL so that calling application is more quickly notified.

2.) Improved handling of scan related timeouts in device driver as well as communication between the DLL and driver for determining the correct timeouts associated with various rates, power cable disconnect scenarios, etc..

3.) Fixed error in **VB** application which was reporting incorrect digital input value directly following a system calibration performed by pressing the "SysCal" button on the main panel

4.) (*Win9x/ME only*) Fixed incorrect reporting of digital input in **LabView** sample application. Due to hardware configuration, digital input value that is read by the board must be XORed with 255.

5.) Version info this release:

VB App: 6,26,2002
LabView App: 6,26,2002
 VC GUI App: 5.21.2002
 VC Console App: 6,19,2001
 VerChk utility: 5,9,2002
DLL: 6,25,2002
DeviceDriver : 6,25,2002
 Hardware: 04

=====

6-3-2002

=====

1.) Various improvements to scan related timeouts and debugging in Device Driver and DLL.

2.) Minor changes in VB sample application related to calling "SP_" style functions in the DLL.

4.) Version info this release:

VB App: 6,3,2002

VC GUI App: 5.21.2002
VC Console App: 6,19,2001
DLL: 6,3,2002
DeviceDriver : 6,3,2002
Hardware: 04

=====

5-13-2002

=====

- 1.) Various improvements made to driver and DLL.
- 2.) New style of VB sample code showing new SP style DLL entry points. Most of what is new is related to threaded style initialization. Please refer to online documentation available from our website for example of using the non-threaded new SP style initialization. We will be adding sample code for the threaded style to the HTML docs at a later date. The threaded style is useful when more than 2 devices are initialized at once, since it allows your application time slicing away from the DLL while the DLL initializes the devices. Without the threaded functionality, the application can appear to "hang" while it is away in the DLL making the initialization function call.
- 3.) A new Version checker utility has been created for checking the DLL/Driver compatibility. Please read "VerChk.txt" located in the application directory.
- 4.) Version info this release:
VB App: 5,13,2002
VC App: 6,19,2001
DLL: 5,13,2002
DeviceDriver : 5,13,2002
Hardware: 04

=====

3-5-2002

=====

- 1.) Improvements made to driver to better handle errors that may be generated by the low level USB hub drivers. Various improvements made to DLL.
- 2.) Fixed problem in VisualBasic sample code that failed to distinguish between duplicate Device IDs in the configuration file.
- 3.) Version info this release:
VB App: 3,4,2002
VC App: 6,19,2001
DLL: 2,27,2002
DeviceDriver : 2,27,2002
Hardware: 04

=====

1-28-2002

=====

- 1.) Various improvements to DLL and device driver. Fixed variable update in VB sample source regarding update of scan number while in multi channel calibration scan mode.
- 2.) Version info this release:
VB App: 1,28,2002
VC App: 6,19,2001
DLL: 1,25,2002
DeviceDriver : 1,25,2002
Hardware: 04

=====

1-17-2002

=====

1.) Various improvements to DLL and device driver related to occasional read errors.

2.) Version info this release:

VB App: 9,10,2001
VC App: 6,19,2001
DLL: 1,17,2002
DeviceDriver : 1,17,2002
Hardware: 04

=====
11-28-2001
=====

1.) Various improvements to DLL and device driver.

2.) Version info this release:

VB App: 9,10,2001
VC App: 6,19,2001
DLL: 11,15,2001
DeviceDriver : 11,15,2001
Hardware: 04

=====
9-10-2001
=====

1.) Changed number of possible devices from 64 to 32. Although this has been the case practically from the beginning of the project, the memory allocation by the DLL and sample applications was never adjusted to show this. That situation has been fixed. Although the sample applications have been adjusted to 32 possible devices in the related arrays, previous applications will continue to work with the new DLL.

2.) Added 3 new entries to LL_USB.INI. They are explained below. These entries should be left alone, unless you experience "microcode buffer wrap" errors.

=====
ScanSystemBoostLevel (default = 0, max = 100)
=====

Use this feature ONLY if you experience "microcode buffer wrap" errors

Improves thread time-slicing handling within the device driver.

Explanation:

Any device driver has the opportunity to "take over" the system for a specified period of time. To do so, it's able to set it's internal thread priority to various levels. However, a mis-behaved driver can still "hog the system" using certain techniques which can prevent other drivers from getting their time slice. We've found that some game programs are notorious for that, which can prevent our driver from getting enough time slices to make the required data transfers from our hardware in the allocated time. This is particularly obvious at high scan rates. A failure to do the required transfers shows as the "microcode buffer wrap" flag being set while scanning. The flag is found as one of the bits (bit-5) of the SO_Status member of the SCAN_OBJECT structure that is used for communication between the DLL and your application. We've incorporated a way around this issue by creating an option to pass a larger packet size request to the "lower USB driver" which has even a higher priority level than our driver is allowed.

Following is an overview of the new feature:

Without using the new "packet boost" feature, our driver reads 32 bytes of data (one packet) at a time from our hardware. That typically translates to 30 bytes of data, and 2 bytes of flags and digital input, which in single channel scan mode, translates to 10 points per read (3-bytes-per-point). However with use of the new packet boost option controlled from the LL_USB.INI file a user can now boost the driver time slicing capability by a factor of up to 100. What this actually does at the driver level, is it allows our driver to make more use of the power of the system driver below the level of ours which has a higher priority level than possible with our own driver. However, this comes with a small price and the price is that your data will be ready for you to read proportionally to the "boost" factor. For example, if you set up a single-channel scan at 1000 Hz without the use of the boost feature, your data will actually be read in 10-point packages and you can

retrieve your data from the DLL with a call every 10 milliseconds. If you set the boost factor to 50, your data will be ready to be read from the DLL every 500 milliseconds, at which time 500 points worth of data will be available. No data is missed, it just takes longer to get the larger packet. Keep this in mind when scanning at slower rates, since if a scan is set to 100Hz and the boost value is set to 100, it will take 10 seconds before one read is complete and that read will contain 10000 points.

NOTE: The ending of the scan process will also take longer as the boost factor is increased.

A good example of an application that can "take control of the system" is the game "Pinball" that ships with WinME and Win2k. That game appears to interact with a low-level driver that completely takes over the system, possibly in order to make possible the smooth graphics that it offers to the user. Running that game on a WinME or Win2k machine while doing a single-channel scan at 1000Hz will usually create a "microcode buffer wrap" within a matter of seconds. However with "ScanSystemBoostLevel" set to 50, such a wrap won't occur.

Advantage: Takes advantage of premium slicing given only to "HUB" type drivers.

Disadvantages:

- 1.) Causes ending of scan to take longer, since each read request by the lower driver has to return with it's packet before for each device before the scanning threads can end.
- 2.) Data retrieval is available at slower rate dependent on the setting of this value. The actual scan rate is unchanged, it's just that each time you read, you will get back much more data, but less frequently. For example in single-channel scan at 1000Hz with boost of 10, you can read the data every 100ms and you'll get 100 points with each read, since the boost causes driver to get 10 packets, of 10 points each, with every read. If you attempt to read every 10ms (the default without use of this flag) your data will be ready only every 10th read.

=====
StackedIrpCntOverride (default = 0, max=40)
=====

Use this feature ONLY if you experience "microcode buffer wrap" errors, and only attempting to correct the problem by setting value above (ScanSystemBoostLevel) to it's maximum (100). Overrides default values calculated by DLL based on scan rate and other variables. The term "IRP" stands for "I/O Request Packet". Our driver communicates with lower drivers (HUB and kernel) in order to communicate with our device. For each read of our device, we pass an "IRP" to the lower driver. Due to processing of the IRPs and data returned from the request, it is usually more efficient to send many at the same time so the lower driver is always busy. However if too many are sent, then time slicing can be taken away from our driver and given to the lower driver.

Advantage: Assures that lower USB driver will always be busy requesting data from our device.

Disadvantages: Too many stacked IRPs can take away from our driver's time slicing and give it to the lower drivers.

=====
DoRecordDrvTimerInfo (default = 0, max=1)
=====

Use this feature ONLY for improving debug file output, when driver time-slicing may be questionable. This should never be set except when debugging time slicing issues in driver which are usually indicated by "microcode buffer wrap" errors. When set, this causes device driver to record timing information and pass it to the DLL which can then write that information to a debug file when the debug file output feature is enabled.

Advantage: Allows display of driver timing issues.

Disadvantages: Puts extra load on driver to record timing information.

3.) **VB Developers only:** Added code to call DLL_EndScan under situation where the scan may have been halted but the device wasn't disconnected. The DLL has an internal thread running while the device is scanning, that continuously sets a flag telling the device driver that the DLL is still functioning. This flag is used for error handling. Typically the DLL resets the flag ever second or so. If the flag has't been set for 6 seconds the device driver automatically ends the scan. Certain system message boxes - for example if you press the ctrl-alt-del keys will take over the system. If such a box is left open for about six seconds the DLL's thread won't get it's time-slice, and the driver will consequentially end the scan. However, previous to the addition of the new code in the VB app, the app would not send an EndScan command under this situation, so the next time a scan was started or any read of the device was done, the first packet would be junk from the device, since it had been left in an unknown state.

Added code to make use of separate thread (timer function) to handle the scan startup for each device. Using the thread allows for processing of data for each device once it's started rather than waiting for all devices to start before retrieving data for any device that has already started. The improvement made by this code is apparent as the devices connected increases. Also added similiar code to handle end-of-scan for each device. Search for text "9-10-01"

Added code to better handle display of errors for individual devices while scanning multiple devices. Search for text "9-10-01".

4.) Version info this release:

VB App:	9,10,2001
VC App:	6,19,2001

DLL: 9,10,2001
DeviceDriver : 9,10,2001
Hardware: 04

=====
8-15-2001
=====

- 1.) Improved DLL initialization and scan startup (less interruption of other applications).
- 2.) Improved registry handling by device driver in regard to keeping track of 301/302 devices add/remove.
- 3.) Improved memory allocation in device driver. Most of the memory required for scanning mode is now allocated when the driver loads each device rather than dynamically when scanning is begun. This should fix the random memory allocation error that has occurred in previous versions on some computer systems.
- 4.) Added additional check for correct version of device driver being loaded. Check is done during initialization. First check is done when DLL is loaded before the application makes any calls into the DLL. That first check checks the version of the driver located on the hard-disk. The second check is done after enumeration as part of the initialization process, which checks the version of the driver loaded in memory. The driver version in memory may not match the version on disk. That situation can occur if for example, you install a new driver by copying it to the disk while any 301/302 is connected. That new driver won't be loaded until all devices are disconnected and then the first one is connected which would cause the first check for driver version to succeed, but the second would find the problem when the DLL_Init() function is called.
- 5.) Increased thread priority (more noticeable by Win2k) in device driver while in scan mode, and further optimized code.
- 6.) **VB Developers only:** Fixed problem with some of the keyboard entry handling in some of the combo boxes on main panel, and modified error handling for initialization. Also improved Scan functions. Search on text "7-27-01", "8-1-01", "8-15-01"

7.) Version info this release:

VB App: 8,15,2001
VC App: 6,19,2001
DLL: 8,13,2001
DeviceDriver : 8,13,2001
Hardware: 04

=====
6-28-2001
=====

- 1.) Improved DLL handling of device add/remove. Also improved load-time error handling within the DLL. The LL_USBLoad.txt file now contains more information concerning errors associated with loading of the DLL such as location of the correct driver version. Also the DLL will now delete any old copies of the whenever it successfully loads.
- 2.) **VB Developers only:** Repaired error associated with handling of channel change that was fixed in previous version. That change had disabled the changing of channels while in the graphics screen. That has been fixed. Search on text "6-28-01".

3.) Version info this release:

VB App: 6,28,2001
VC App: 6,19,2001
DLL: 6,28,2001
DeviceDriver : 6,11,2001
Hardware: 04

=====
6-21-2001
=====

NOTE: Includes changes for special CD release 6-20-2001 with a couple of additions

- 1.) See additions to the "extended explanation" of the DLL_Init() function (comment #7) in "API_USB.DOC".

- 2.) See updated "INSTALL.DOC" - There is now only one .INF file for installing to Win98/WinME/Win2k
- 3.) Added DLL function `DLL_SetSpecialDebugOptions()`. See `API_USB.DOC` for usage instructions.
- 4.) The DLL now maintains a copy of the application's device ID list rather than relying on the pointer to the app's list as was previously the case. I found that VisualBasic does not always unload the DLL when the Windows API function `FreeLibrary()` is called - for example if any of the DLL's function calls have been made since the DLL was first loaded. It apparently intercepts that call, and then it waits to unload the DLL until the applications is just ready to unload. It turns out that that occurs after the memory used by the application has been freed and therefore the original application device ID pointer passed to the DLL is no longer valid. The DLL when it unloads, attempts to do any cleanup that the application may not have done and to do so, needs a list of the devices the application was using.
- 5.) `DLL_ReInitOneDev()` as been fixed. It had lost some functionality in the course of some of the previous changes to the DLL.
- 6.) **VC Developers only:** Search for text "6-18-01" and "6-19-01". Modified sample application to fix error in starting 2-channel scan. Also fixed keyboard handling for the scan configuration screen. See `API_USB.DOC` for information on using new scan modes and features.
- 7.) **VB Developers only:** Search for text "6-20-01" and "6-21-01". Modified sample application to fix error in starting 2-channel scan. Also fixed keyboard handling for the scan configuration screen, and a few other small problems
- 8.) Version info this release:

VB App:	6,21,2001 (6,20,2001 special release)
VC App:	6,19,2001
DLL:	6,21,2001 (6,19,2001 special release)
DeviceDriver :	6,11,2001
Hardware:	04

=====

5-22-2001

=====

- 1.) Modified DLL to improve cable swapping between devices. and handling of internal scanning threads.
- 2.) **VC Developers only:** Search for text "5-21-01" and "5-22-01". Modified sample application to improve logging of scan data to file. Fixed error associated with logging of data with averaging set in cal-scan mode. Placed reading of scan data within a timer function. Since VB assigns a lower priority to a timer function than it does to it's main thread, this reduced the CPU usage shown in the Win98 CPU usage meter.
- 3.) Version info this release:

VB App:	5,22,2001
VC App:	5,2,2001
DLL:	5,21,2001
DeviceDriver :	5,2,2001
Hardware:	04

=====

5-7-2001

=====

NOTE: New Application, configurartion file, and documentation filenames.

NOTE: Previous DOCs archived in "old-docs.zip" which is included in app souce code dir.

- 1.) Modified DLL, Devised driver, and both sample applications. Modified hardware to add additional channels to 301 and option for digital input with each scan data point for 301 and 302. No hardware version number change for these new features.
- 2.) Added `DLL_ReadScanDataWithDigin()` function. To make use of this scanning function requires a newer microcontroller with a device ID number greater than 154. See `API_USB.doc` for function prototype and instructions for use.
- 3.) **VB Developers only:** Search for text "5-2-01" and "5-8-01".

Only one sample application and VB source code now used for both the Model-301 and Model-302. It's source code filenames have been changed to reflect this. It's executable file has been renamed, "LL_USB.exe". The main difference between the two models now, is the 302 has analog output capability and 8 scannable channels and the 301 has no analog output and only 4 scannable channels (2 with Dev IDs less than 155).

FOLLOWING ELIMINATED

'4-18-01 frmLL_USB.btnSelScanChans.Visible = True
'4-18-01 frmLL_USB.ShapeSelScanChans.Visible = True
'4-18-01 frmLL_USB.LabelSelScanChans.Visible = True

Graphics modified. Handling of lists and panel controls when a device is removed while scanning has been improved. Scan logfile output improved and now includes digital input.

4.) **VC Developers only:** Search for text "4-24-01". Changed application handling of channel change screen drawing during scan. Also changed some of the variable names related to model number.

5.) FOLLOWING SCAN STRUCTURE MEMBERS RENAMED

from: SO_iCompletedScans As Long
to : SO_iTotalPointsReadByDrvr

from: SO_iScansInBuffer As Long
to: SO_iPointsInBuffer

6.) When starting a multi-channel scan with the model 301 set the SO_bScanArg variable to the number of channels you wish to scan (when using DevIDs > 154).

7.) Only one configuration, "**LL_USB.CFG**" now used **for both the Model-301 and Model-302** sample applications. Configuration file lines have been rearranged to add the default model number. Whichever Model number is entered there is the configuration that will be used when the main panel loads (VisualBasic) and initializes it's variables and controls. There is a checkbox on main panel (VisualBasic) to change to a different model number. The one that is checked (VisualBasic) when main panel is loaded comes from the 3rd line in the configuration file. The additional Model 302 features haven't yet been integrated into the VisualC++ sample application.

8.) New name used for the debug file output created by the DLL. New name is M30x_DBG where 'x' is actually the letter 'x' and not a number, since same naming convention is used for both the model 301 and the model 302.

9.) Version info this release:

VB App: 5,8,2001
VC App: 5,2,2001
DLL: 5,2,2001
DeviceDriver : 5,2,2001
Hardware: 04