TI-83 Plus
SILVER EDITION

Addendum to the TI-83 Plus Developer Guide

Beta Version .02

IMPORTANT INFORMATION

Texas Instruments makes no warranty, either expressed or implied, including but not limited to any implied warranties of merchantability and fitness for a particular purpose, regarding any programs or book materials and makes such materials available solely on an "as-is" basis.

In no event shall Texas Instruments be liable to anyone for special, collateral, incidental, or consequential damages in connection with or arising out of the purchase or use of these materials, and the sole and exclusive liability of Texas Instruments, regardless of the form of action, shall not exceed the purchase price of this program. Moreover, Texas Instruments shall not be liable for any claim of any kind whatsoever against the use of these materials by any other party.

TI-83 Plus
SILVER EDITION

The following information is provided as an addendum to the TI-83 Plus Developer Guide Version 1.0. This text provides additional information for developing applications and assembly programs for the TI-83 Plus *SILVER EDITION*. Specific information deals with the Flash ROM size and processor speed changes over the TI-83 Plus graphing calculator platform.

OVERVIEW

The TI-83 Plus *SILVER EDITION* has the capability to load up to 94 Flash Apps and has a clock speed that is more than twice as fast as the standard TI-83 Plus. For compatibility, the TI-83 Plus *SILVER EDITION* runs programs and applications at the standard TI-83 Plus speed. The TI-83 Plus *SILVER EDITION* also contains a hardware assist to speed up link communication.

Flash Apps can utilize new Operating System (OS) calls or header settings to take advantage of the increased speed. Assembly programs can also use the OS calls to set speed. The new calls will not be recognized by older OSs so it is important for the developer to check OS versions when making these calls.

FLASH ROM STRUCTURE

The TI-83 Plus *SILVER EDITION* Flash ROM is composed of 2048K (2M) bytes divided into 128 pages, each of which is 16K bytes in size. The structure is generally the same as the TI-83 Plus except for the inclusion of 96 additional 16K pages (24 additional 64K Sectors). The TI-83 Plus *SILVER EDITION* can store up to 94 Apps in pages 0Ch – 69h.

The TI-83 Plus Flash structure chart, Figure 2.5 in the TI-83 Plus Developer Guide Version 1.0, is correct up to page 14h; at that point, the TI-83 Plus *SILVER EDITION* includes more data pages. The TI-83 Plus *SILVER EDITION* also has the Operating System residing at the high 8 pages of Flash, 78h . . . 7Fh. The TI-83 Plus high memory is 18h . . . 1Fh.

	Addr	Page(s)	Size	Notes
OS	00h – 07h	00 – 07	128K (8 pages)	Same as TI-83 Plus
SWAP/USER DATA	08h – 0Bh	08 – 11	64K (4 pages)	Same as TI-83 Plus
SWAP/USER APPS/DATA	0Ch – 0Fh	12 – 15	64K (4 pages)	Same as TI-83 Plus
USER APPS/DATA	10h – 13h	16 – 19	64K (4 pages)	Same as TI-83 Plus
USER APPS/DATA	14h – 67h	20 – 103	1344K (84 pages)	Many more user pages
USER APPS/DATA	68h – 69h	104 – 105	32K (2 pages)	Pages 14h, 15h on TI-83 Plus
CERTIFICATE LIST	6Ah – 6Bh	106 – 107	32K (2 pages)	Pages 16h, 17h on TI-83 Plus
FUTURE OS USE	6Ch – 77h	108 – 119	192K (12 pages)	
OS	78h – 7Fh	120 – 127	32K (8 pages)	Pages 18h . . . 1Fh on TI-83 Plus

Legend

SWAP and/or User APPS Area
Update System (OS) Area
Fixed Area — changeable only by TI

Because of the additional Flash memory, the first App loaded will be placed at page 69h (105 decimal) and downward from there (versus 15h for the TI-83 Plus).

ENTRY POINTS, VARIABLE TYPES, AND SYSTEM FLAGS

The TI-83 Plus *SILVER EDITION* uses the same entry points, variable types, and system flags as the TI-83 Plus.

ADDITIONAL B_CALLS FOR DETERMINING OS AND SPEED

The TI-83 Plus *SILVER EDITION* and TI-83 Plus OS version 1.13 includes new B_CALL entry points that allow an application to set the processor speed (**SetExSpeed**) and to determine the current speed and OS (**GetSysInfo**).

The ti83plus.inc has the following entries:

```
SetExSpeed    EQU         50BFh
GetSysInfo    EQU         50DDh
NzIf83Plus    EQU         50E0h
```

See the entry point documentation in Appendix A for additional information.

ADDITIONAL HEADER TAGS FOR SETTING SPEED

By default, an application will run at the TI-83 Plus Speed unless the App informs the OS to allow it to run faster. The entry point **SetExSpeed** controls specific locations in the code.

An optional method is to set a tag in the header that informs the OS that the App can run at a certain level. The default level is 0 and the TI-83 Plus *SILVER EDITION* can be set to level 1.

The APP_HW_LEVEL tag looks as follows:

```
            DB          080h,0A1h         ; Field: App level
            DB          001h              ; Highest HW level = 1
```

If this field is missing or is set to HW level 0 (either DB 080h,0A0h or DB 080h,0A1h,000h), then the Operating System executes the App in slow mode (6MHz) to execute the App. This field is ignored on the TI-83 Plus.

```
; This is the application header definition area required for all Apps.

            DB          080h, 00Fh              ; Field: Program length
            DB          000h, 000h, 000h, 000h  ; Length = 0 (N/A for unsigned
                                                ; Apps)
; ------------------------------------------------------------------------------
; * * * App types * * *
; This example uses shareware type for signing and the simulator.
; Replace with the commented developer ID for debugging on the calculator.
; Note the header will need to be adjusted depending on the number of bytes in
; this type field.
; ------------------------------------------------------------------------------
            DB          080h, 012h              ; Field: Program type (2 byte)
            DB          001h, 004h              ; Type = Shareware, TI-83 Plus
;                                               -------------------------------
;                                               ; Use with 5 or 6 character ID
;           DB          080h, 013h              ; Field: Program type (3 byte)
;           DB          001h, 07Fh, 004h        ; Dev. ID = "17F04"
; ------------------------------------------------------------------------------
            DB          080h, 021h              ; Field: App ID
            DB          001h                    ; Id = 1
            DB          080h, 031h              ; Field: App Build
            DB          001h                    ; Build = 1
; ------------------------------------------------------------------------------
; -- App Name is to be 8 characters in size --
            DB          080h, 048h              ; Field: App Name
            DB          "TEMP    "              ; Name = "TEMP    "
            DB          080h, 081h              ; Field: App Pages
            DB          001h                    ; App Pages = 1
            DB          080h, 090h              ; No default splash screen
; ------------------------------------------------------------------------------
            DB          080h, 0A1h              ; Field: App level
            DB          001h                    ; Highest HW level = 1
; ------------------------------------------------------------------------------
            DB          003h, 026h, 009h, 004h  ; Field: Date stamp =
```

```
                DB          005h, 0D4h, 062h, 000h  ; 2/7/2000
                DB          002h, 00Dh, 040h        ; Dummy encrypted TI date
                DB          055h, 073h, 021h, 0E3h  ; stamp signature
                DB          03Bh, 081h, 022h, 017h
                DB          02Dh, 0D2h, 0D3h, 018h
                DB          093h, 063h, 078h, 0A6h
                DB          0A2h, 006h, 05Ch, 071h
                DB          0C0h, 031h, 0E5h, 098h
                DB          0DEh, 06Dh, 039h, 03Ch
                DB          0F8h, 035h, 0E0h, 0A7h
                DB          00Fh, 092h, 0A5h, 037h
                DB          068h, 0F3h, 040h, 019h
                DB          06Eh, 0CAh, 02Fh, 064h
                DB          0E9h, 0AAh, 0CFh, 0C9h
                DB          035h, 039h, 0C0h, 043h
                DB          05Bh, 0D3h, 037h, 086h
                DB          041h, 0E2h, 001h, 090h
; --------------------------------------------------------------------------
                DB          080h, 07Fh              ; Field: Program Image length
                DB          000h, 000h, 000h, 000h  ; Length = 0, N/A
; --------------------------------------------------------------------------
; End of required Header.
; To allow for growth of header fields during the signing process, the
; following pad bytes are needed. Adjust as needed.
                DB          0, 0, 0, 0              ; Reserved Pad
                DB          0, 0, 0, 0              ; Reserved Pad
                DB          0, 0, 0, 0              ; Reserved Pad
                DB          0                       ; Adjusted pad for level type
; --------------------------------------------------------------------------
; Adjust header according to application type.
; This example uses shareware type (2-bytes).
; Replace with the commented pads if using a 3-byte developer ID.
                DB          0, 0, 0, 0              ; Reserved bytes with pad for
                                                    ; 2-byte Program Type field
;               DB          0, 0, 0                 ; Reserved bytes with pad for
                                                    ; 3-byte Program Type field
;
; End of 128 byte application header
; --------------------------------------------------------------------------
;
; Execution starts here.
```

POSSIBLE DISPLAY PROBLEMS RUNNING FAST ON THE SILVER EDITION

There are problems that can occur if an application that was written for the TI-83 Plus is run at fast speed on the TI-83 Plus *Silver Edition*.

The screen may not display correctly if the app writes directly to the display and has its own version of LCD_BUSY instead of calling the system routine. The LCD has a delay requirement of approximately 10us between operations and using the old LCD_BUSY delay at the faster speed will be less than this.

There are three options for solving this problem.

- Triple or quadruple the delay time of the in-line code. This will solve the problem, but it may reoccur if another even faster version is produced.
- Do B_CALL LCD_BUSY. This is guaranteed to work, but may slow down a display intensive app.
- Use a CALL LCD_BUSY_QUICK, where LCD_BUSY_QUICK is equated to 000Bh. This is a new entry point that does not require the system overhead of a B_CALL. This call also works on earlier TI-83 Plus versions, but runs slightly faster than 10us and modifies the z/nz Status Flag. To use this on all versions, wrap it in another routine that saves and restores the flag register:

```
PUSH        AF              ; 11 states
CALL        LCD_BUSY_QUICK  ; 17 states + 30 states (on TI-83 Plus)
POP         AF              ; 10 states
```

This will ensure that the routine runs on both the TI-83 Plus and the TI-83 Plus *Silver Edition* with minimal additional time delays.

ACCESSING THE DBUS FROM AN APP

The TI-83 Plus *SILVER EDITION* is equipped with a DBUS assist unit in the hardware. This hardware assist will greatly speed up Apps that do significant DBUS activity, but requires additional care on the part of the App programmer. When used incorrectly, the DBUS assist will cause the App to suffer from timeout errors even though the data is sent/received correctly. The paragraphs below provide several use scenarios, possible problems encountered, and possible fixes.

Monitoring DBUS Lines to Detect DBUS Activity

Frequently, an App will need to wait for a byte to be sent from another calculator while the App is in a loop doing something else. On the TI-83 Plus, it was standard practice to input the state of the DBUS lines on each pass through the loop and then to make a B_CALL to **RecAByte** ONLY when one of the DBUS lines was active (a low).

With the hardware assist, this process will fail. The failure is due to the fact that the hardware assist will complete the handshake when the DBUS line goes active by reading in the incoming byte and storing the byte in a hardware buffer. Thus the DBUS lines will return to an inactive state in a few microseconds and the software will never detect any activity.

The corrective action here is to first determine the type of hardware that the App is being executed on by calling the **GetSysInfo** utility. If the hardware type indicates a TI-83 Plus *SILVER EDITION* unit (e.g., Bit 03 is set), then activity on the DBUS lines must be determined by executing the following code:

```
IN          (0x09), A
AND         0x38
```

If the ‘nz’ bit is set, the DBUS hardware assist has stored a byte for retrieval OR IS IN THE PROCESS OF RECEIVING A BYTE AT THE PRESENT TIME. This is the only reliable way of determining activity on the DBUS on the TI-83 Plus *SILVER EDITION* hardware.

Sending DBUS Data/Error States by Toggling the DBUS Lines

Frequently Apps will attempt to flag an error to the DBUS by taking both DBUS lines low. Similarly, Apps may attempt to toggle the DBUS lines from within the App rather than by invoking the **RecAByte** and **SendAByte** routines. These actions will cause the DBUS hardware assist to reset each time either (or both) of the DBUS lines are placed into the low state. If the DBUS assist hardware is in the middle of a transmission when the App toggles the DBUS line, the transaction is aborted and the byte being sent or received is lost.

Unexpected Error Conditions

If the DBUS hardware assist unit is sending data to a DBUS device and that device asserts both DBUS lines low, the DBUS assist will abort the send with an error condition. The indicated error will be TIMEOUT, as the hardware will only know that it could not send the byte in the 2 seconds allotted.

Summary

In general, it is always safe to call **RecAByte** and **SendAByte**. Direct access to the DBUS lines is discouraged as it may cause communication problems.

GetSysInfo

Category: Utility

Description: Return nine bytes of system information, including current speed.

Input:

Registers: HL = RAM location of where to put system information

Flags: None

Others: None

Outputs:

Registers: None

Flags: None

Others: (HL) . . . (HL + 8) set depending on system information.

Registers destroyed: AF, BC, HL

Remarks: Note that this B_CALL is not available on TI-83 Plus version 1.12 and earlier. The calling routine needs to check the software version before performing the B_CALL.

This routine returns nine bytes representing various aspects of system operation:

Byte00 = Boot Code Revision # (Major)
01 = Boot Code Revision # (Minor)
02 = Hardware Revision # (00 if TI-83 Plus, NZ if not)
03 = Lsn = Current Speed
03 = Bit 4 = reset if TI-83 Plus, set if TI-83 Plus *SILVER EDITION*
04 = Device code default
05 = 0 = reserved for future use
06 = 0 = reserved for future use
07 = 0 = reserved for future use
08 = 0 = reserved for future use

(continued)

GetSysInfo *(continued)*

Example: Determine if are running fast or slow:

```
                B_CALL      getBaseVer      ; OS version in (A, B)
                CP          2               ; major version
                JR          NC,above112     ; if 2.x than > 1.12
                CP          1
                JR          NZ,MustBeSlow   ; if 0.x then < 1.12
                LD          A,B             ; major version is 1
                CP          13              ; minor version
                JR          NC,above112     ; C if minor version < 13
MustBeSlow:
                XOR         A               ; set Z to show slow
                JR          Done
above112:                                   ; later than version 1.12
                LD          HL,OP1
                B_CALL      GetSysInfo
                LD          A,OP1+3
                AND         0Fh
Done:
                .
                .
                .
```

NzIf83Plus

Category: Utility

Description: Return status if calculator is a TI-83 Plus or not.

Input:

Registers: None

Flags: None

Others: None

Outputs:

Registers: None

Flags: NZ = Status if calculator is a TI-83 Plus
Z = Status if calculator is a TI-83 Plus *SILVER EDITION*

Others: None

Registers destroyed: None

Remarks: Note that this B_CALL is not available on Cerebrus version 1.12 and earlier. The calling routine needs to check the software version before performing the B_CALL.

This routine is not as intrusive as GetSysInfo if all you need to know is if the calculator is an earlier edition of the TI-83 Plus.

Example: Return NZ if are running on a TI-83 Plus.

```
              B_CALL      getBaseVer      ; OS version in (A, B)
              CP          1               ; major version
              JR          C,MustBe83Plus  ; if 0.x than < 1.13
              JR          NZ,Above112     ; if 2.x then > 1.12
              LD          A,B             ; major version is 1
              CP          13              ; minor version
              JR          NC,above112     ; C if minor version < 13
MustBe83Plus:
              RET                         ; NZ status

above112:                                 ; later than version 1.12
              B_CALL      NzIf83Plus      ; NZ if original TI-83 Plus
              RET
```

SetExSpeed

Category: Utility

Description: Set execution speed to fast or slow.

Input:

Registers: A = 0 to set slow speed (6MHz)
A = 1 to set 15MHz
A = FF to set Fastest Future speed

Flags: None

Others: None

Outputs:

Registers: None

Flags: None

Others: None

Registers destroyed: Flag register modified.

Remarks: Note that this B_CALL is not available on TI-83 Plus version 1.12 and earlier. The calling routine needs to check the software version before performing the B_CALL.

This routine can be called on a TI-83 Plus unit running software version 1.13 or higher, but will not affect the operating speed of that unit.

The operating system will set the speed back to fast once the app or assembly program returns regardless of any settings made. An exception to this is that the error handler will leave the speed setting as is in case a goto is desired.

Some system routines such as the IO utilities may set slow speed for certain operations. These routines will restore the current speed upon completion. Other routines, such as the JForceCmd instruction force the fast clock. Normally an app will not execute these routines except upon completion.

(continued)

SetExSpeed *(continued)*

Example: Determine if the app is running on operating system version 1.13 or higher and if so, run at the fast clock frequency:

```
                B_CALL      getBaseVer      ; operating system version
                                            ; in (A, B)
                CP          2               ; major version
                JR          NC,above112     ; if 2.x then > 1.12
                CP          1               ; if 0.x then < 1.12
                JR          NZ,below112     ; major version is 1
                LD          A,B             ; minor version
                CP          13              ; C if minor version < 13
                JR          C,below112      ; later than version 1.12
above112:
                LD          A,0FFh          ; set fastest speed possible
                B_CALL      SetExSpeed

below112:
.
.
.
```