TRANSWARP GS -
avec mes amitiés
D. Nellhaus

APPENDIX B

Programmer's Reference

Firmware Addresses

The TransWarp GS Firmware has a jump table located at **$BCFF00** that can be used to identify and configure the TransWarp GS hardware from within an application.

$BC/FF00-FF03 **TransWarpID** (4 bytes)

The value at this location can be used to identify the presence of the TWGS card. The value is the positive-ASCII string 'TWGS' ($54 57 47 53).

$BC/FF04-FF07 **TransWarpID2** (4 bytes)

These four additional ID bytes are not used and are set to the positive-ASCII string 'SMJS' ($53 4D 4A 53).

Firmware Subroutines

The following are subroutines within the TWGS' firmware. All subroutines are called in full native mode (e=0, m=0, x=0) via a JSL instruction. Assume that the contents of any register which is not documented as returning a result are destroyed.

$BC/FF08-FF0B **GetTWInfo** BC FA C4

Inputs: none
Outputs: A: Hardware features
X: TransWarp GS version/revision number

GetTWInfo returns the version/revision number of the TWGS as well as information about the features available.

$BC/FF28-FF2B **GetCurISpeed** BC FB 40

Inputs: none
Outputs: X: speed index

GetCurISpeed returns the speed index for the clock rate at which the TransWarp GS is currently running.

$BC/FF2C-FF2F **SetCurISpeed** FB 1C

Inputs: X: speed index
Outputs: none

SetCurISpeed sets the TransWarp GS to run at the clock rate specified by the speed index.

$BC/FF30-FF33 **FlushCache**

Inputs: none
Outputs: none

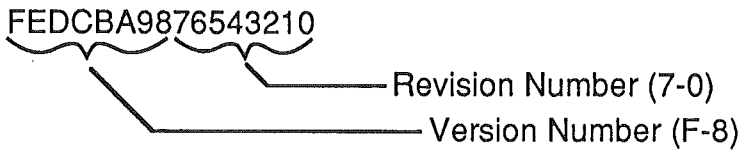
FlushCache empties the contents of the cache RAM on the TWGS. This assures that the first subsequent read to any memory location will not come from the TransWarp GS's cache memory. Note that if GetTWInfo returns with the 'hardware cache flush' bit cleared, then the flush will be done by the firmware which will take a relatively long time (usually less than .5 seconds, depending on the cache size).

$BC/FF34-FF37 **DisableIRQLogic**

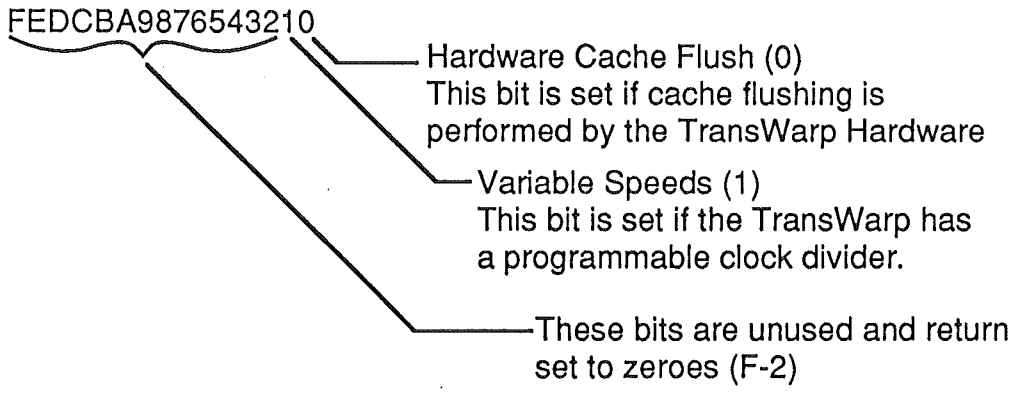
Inputs: none
Outputs: none

DisableIRQLogic turns off the TWGS circuits that control the speed while the CPU has interrupts disabled. Normally the TWGS hardware will slow the CPU down to the speed set in the GS's speed register at $E0C036 whenever interrupts are disabled by the CPU. This assures that timing-critical routines (many of which disable interrupts while they're-executing) run at the proper speed.

Version Word:



Features Bits:



$BC/FF0C-FF0F **ResetTW** BC FACB

Inputs: none
Outputs: none

ResetTW restores the TWGS hardware configuration to the values set by the user in the TWGS control panel.

$BC/FF10-FF13 **GetMaxSpeed** BC FAD3

Inputs: none
Outputs: A: maximum clock rate (in KHz)

GetMaxSpeed returns the fastest speed at which the TransWarp GS can be set to run.

$BC/FF14-FF17 **GetNumISpeed** BC FA D7

Inputs: none
Outputs: X: number of discreet speeds available

GetNumISpeed returns the number of discreet speeds at which the TWGS can be set to run

$BC/FF18-FF1B **Freq2Index**

Inputs: A: clock rate (in KHz)
Outputs: X: speed index

Freq2Index returns the nearest speed index whose clock rate is greater than or equal to the specified clock rate. If there is not a speed index available which is greater than the specified clock rate then the maximum speed index is returned (which will be one less than the value that is returned by GetNumISpeed).

$BC/FF1C-FF1F **Index2Freq**

Inputs: X: speed index
Outputs: A: clock rate (in KHz)

Index2Freq returns the clock rate for the specified speed index. The speed index is a value from zero to one less than the value returned by GetNumISpeed. Speed index values of zero or one represent the GS' native speeds of slow (1 MHz) and fast (2.6MHz), respectively. An illegal speed index will always return a clock rate of zero in A.

$BC/FF20-FF23 **GetCurSpeed** BC FB62

Inputs: none
Outputs: A: current clock rate (in KHz)

GetCurSpeed returns the clock rate at which the TransWarp GS is currently running.

$BC/FF24-FF27 **SetCurSpeed** BC FB74

Inputs: A: clock rate (in KHz)
Outputs: A: actual clock rate (in KHz)

SetCurSpeed sets the TWGS to run at the specified clock rate. The actual clock rate that the TWGS was set to run at is returned in case the requested speed is not available. The actual clock rate will be the nearest available clock rate that is greater than or equal to the requested value. However, when the requested clock rate is greater than the maximum clock rate, the value returned will be the maximum clock rate.

There are times, however, when programs disable interrupts to allow themselves to run at the fastest possible speed. DisableIRQLogic and EnableIRQLogic (below) allow a program to configure the TWGS for maximum performance and compatibility. Note that the default setting for the IRQ slowdown logic is set by the user through the 'AppleTalk/IRQ' option in the TWGS control panel (on = enabled).

$BC/FF38-FF3B **EnableIRQLogic**

Inputs: none
Outputs: none

EnableIRQLogic turns on the TransWarp GS's circuits that control the speed while the CPU has interrupts disabled. See the description of DisableIRQLogic (above) for an explanation of how this call is used.

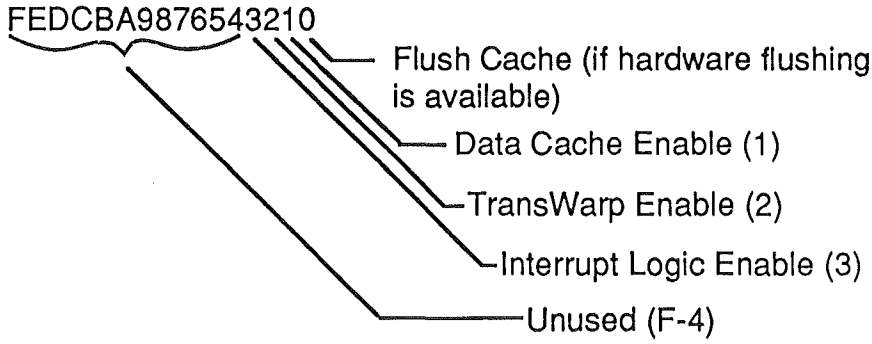
$BC/FF3C-FF3F **GetTWConfig**

Inputs: none
Outputs: A: current configuration register setting

GetTWConfig returns the current setting of the TWGS configuration register.

❖ *Note:* The meaning of the Config register bits may change in future versions and should not be relied upon for determining the state of the TWGS.

Configuration register bits:



$BC/FF40-FF43 **SetTWConfig**

Inputs: A: configuration register setting
Outputs: none

SetTWConfig sets the TWGS configuration register to the specified value. This call should not be used to change any of the configuration register settings. The purpose of the GetTWConfig and SetTWConfig routines is to allow applications to save and restore the TWGS configuration. The other routines in this interface should be used to change the configuration settings.

$BC/FF44-FF47 **GetCacheSize**

Inputs: none
Outputs: A: size of cache

Returns the size of the TransWarp GS cache in 1K multiples (i.e. 4 = 4096 bytes).

$BC/FF48-FF4B **EnableDataCache**

Inputs: none
Outputs: none

EnableDataCache will allow CPU data fetches to come from the TWGS' cache. This is the default setting for the data cache enable.

$BC/FF4C-FF4F **DisableDataCache**

Inputs: none
Outputs: none

DisableDataCache will prevent CPU data fetches from coming from the TransWarp GS' cache. Opcode and operand fetches will still be cached as they normally are.

Programming examples:

These examples demonstrate some of the typical ways in which the firmware interface can be used. All of the examples are written using APW conventions.

modifié pour

Example 1: Identifying the TransWarp GS

```
            lda   >$BCFF00
            cmp   #'TW'      -->  Cmp #'WT'
            bne   noTransWarp
            lda   >$BCFF02
            cmp   #'GS'      -->  CMP #'SG'
            bne   noTransWarp
            ...                   TransWarp-dependent code
                                  goes here

noTransWarp ...                   there's not a TransWarp GS
                                  present
```

Example 2: Preserving the current configuration settings

```
            jsl   GetTWConfig     get the configuration
            pha                   ...and save it

            ...

            pla                   get back the saved
                                  configuration
            jsl   SetTWConfig     ...and restore it
```

Example 3: Setting the TransWarp GS to run at its fastest speed

```
            jsl   GetMaxSpeed     get the maximum speed
            jsl   SetCurSpeed     ...and set the TransWarp to it
```

Example 4: Checking for the availability of a specific speed

```
            lda   #7000           Is 7MHz available?
            jsl   Freq2Index      convert it to an index
            jsl   Index2Freq      ...and back again
            cmp   #7000           was it available?
            bne   notAvailable

            ...                   the desired speed was
                                  available

notAvailable ...                  the desired speed was not
                                  available
```

Example 5: Finding out what speeds are available

```
                jsl     GetNumISpeeds   get the number of discreet
                                        speeds
                phx                     ...and save it

                ldx     #0              start with the first speed
nextSpeed       jsl     Index2Freq      get the clock rate

                ...                     the clock rate is in A

                inx                     increment to the next speed
                txa
                cmp     1,s             are we done?
                bcc     nextSpeed

                pla                     clean off the stack
```