Pluto

Pluto 5 Development Kit Software Release Notes

Document No. 80-17008 Issue 36 HEBER Ltd.

Current Issue :- Issue 36, 5th February 2007

Previous Issue :- Issue 35, 3rd February 2006

HEBER LIMITED
Belvedere Mill
Chalford
Stroud
Gloucestershire
GL6 8NT
England

Tel +44 (0) 1453 886000
Fax +44 (0) 1453 885013
Email support@heber.co.uk
Web site http://www.heber.co.uk

CONTENTS

This page intentionally left blank.

1 INTRODUCTION

The software that is installed with the Pluto 5 Development Kit is the current version at the time that the product was released. Periodically the Heber Pluto 5 software will be revised. The updated software can be downloaded from the Heber web site www.heber.co.uk.

It is recommended that you check the web site and download the latest versions of the software drivers before starting game development.
If you experience problems with the software during game development please check the web site to ensure that you are using the latest versions of the software.

This document provides information regarding changes made for each version of the software. This document is updated after each release with a brief description of the problem that was identified. From time to time, when a better description is required Heber will include special notes on the problem, side effects or code description.

For any further information or to report any problems with the software contact Heber Customer Support: support@heber.co.uk

2 HARDWARE LIBRARY (81-16082)

2.1 Issue 17 (5/10/01)

- Calypso 16 support added.

2.2 Issue 18 (14/3/02)

- IDE support for Calypso 16 added

2.3 Issue 19 (19/4/02)

- Calypso 16 Driver: Speed improvements to *load_image()*

2.4 Issue 20 (25/4/02)

- ATA Driver added

2.5 Issue 21 (7/2/03)

- Calypso 16 Driver: Function *create_mask()* added to support loading Compact Flash sprites.

2.6 Issue 22 (26/6/03)

- **Major Release: Calypso 16 UART Driver added.**

2.7 Issue 23 (21/10/03)

- Serial Drivers: Various bug fixes.

2.8 Issue 24 (19/02/04)

- I2C Driver: Fix related to internal I2C for security PIC **(Note 5.3)**

2.9 Issue 25 (01/03/04)

- Low-level video driver: Function modified: create_mask(). Fixes random dot screen corruption when loading transparent sprites from Compact Flash

2.10 Issue 26 (28/04/04)

- **Major Release:** Enhanced Calypso Driver Added. **(Refer to Pluto 5 Software User Manual).**

2.11 Issue 27 (07/05/04)

- Heber Internal Release – No functional changes.

2.12 Issue 28 (10/05/04)

- Enhanced Calypso driver – Unicode character support added. **(Refer to Pluto 5 Software User Manual).**
- Enhanced Calypso driver – Animation support added. **(Refer to Pluto 5 Software User Manual).**

2.13 Issue 29 (26/05/04)

- 68340 Serial driver: Function ClearRx() fixed to re-enable the receive interrupt.

2.14 Issue 30 (09/06/04)

- I2C Driver. Timing issue for PIC to come back to start of state machine.

2.15 Issue 31 (07/07/04)

- Heber Internal release.

2.16 Issue 32 (16/07/04)

- Sound Driver: Work-around to fix problem of system crash caused by corruption of tune index.

2.17 Issue 33 (16/07/04)

- Pluto5 initialisation and low level services:

CRITICAL BUG FIX to prevent incorrect clearing of battery-backed System RAM and Identity RAM, following a bus-error exception.

When the platform crashes for any reason, the interrupt subroutine was running a “BGND” instruction.
When a debugger is plugged into the board, the BGND instruction stops the code from running and a user can then look at your stack pointer and try to debug their code.

When the code is running from EPROM, there is no debugger to intercept this instruction. The 68340, sees the “BGND” instruction and sends another interrupt “illegal_inst” where a BGND instruction will be run again.

At this point, this function will call itself recursively until the stack overflows forcing the platform to restarts with a different checksum and identity causing all memory to be erased

2.18 Issue 34 (16/07/04)

- Sound Driver: fix bug causing *tunes to play twice*. Problem caused by tune index out-by-one error.

2.19 Issue 35 (05/08/04)

- Output Driver: fix bug causing corruption of *sound driver* variables. This caused errors when playing sequences of tunes.

2.20 Issue 36 (06/10/04)

- **Major Release:** Duart bug fix **(Note 5.6)**.

2.21 Issue 37 (22/10/04)

- Advanced video driver: Bug fix when writing a Unicode string to video memory (CalypsoPutUnicodeStr).
- Ata driver improvement: Improvement to the driver in order to avoid delays in data transfer caused by some compact flash card.

2.22 Issue 38 (10/11/04)

- Ata driver: A variable was not initialised correctly which caused the MountFat32 function not to initialize when program was run from EPROM.

2.23 Issue 39 (24/01/05)

- Duart driver Bug fix when working at 38400 bps

There are two tables of baud rates for the DUART. The speed tables are given in the Philips Semiconductors SCC68692 datasheetp16

A restriction of the duart is that both channels must use the same table at once. There are still problematic speed combinations now but we have modified our driver to be consistent with the datasheet.

2.24 Issue 40 (11/01/06)

Advanced Video driver: At the end of an animation, the last sprite is lost. Now, when the animation finishes, background and final printing sprite are visible on screen.

2.25 Issue 41 (11/01/06)

Vfd2 driver: A problem was identified when using the vfd2 driver. Problem fixed but the file config.c, requires the definition of the two Vfd’s systematically.

3 PERIPHERALS LIBRARY (81-16083)

3.1 Issue 12 (10/12/01)

- Calypso 16 driver added

3.2 Issue 13 (14/3/02)

- Calypso 16 advanced driver added

3.3 Issue 14 (4/4/02)

- Calypso 16: Speed improvements to *PutText()*
- Calypso 16: BUG FIX to out-by-one *PutSprite()* height parameter

3.4 Issue 15 (19/4/02)

- Calypso 16: BUG FIX Image loading

3.5 Issue 16 (22/4/02)

- Calypso 16: *cmd_in_progress* semaphore added.

3.6 Issue 17 (3/5/02)

- CD-ROM Driver: Modification to support 52X drives

3.7 Issue 18 (7/5/02)

- Calypso 16: *ClearLayer()* function added.

3.8 Issue 19 (9/5/02)

- Calypso 16: Cremson FIFO empty check added.

3.9 Issue 20 (22/11/02)

- Cirrus 5429 Driver: BUG FIX to Palette initialisation
- Calypso 16: BUG FIX for Cremson FIFO

3.10 Issue 21 (27/11/02)

- Calypso 16: BUG FIX for Window Layer initialisation**.**

3.11 Issue 22 (29/11/02)

- Calypso 16: New Palette Functions added

3.12 Issue 23 (21/1/03)

- Calypso 16: *CremsonByteOrder()* function added **(Note 5.1)**

3.13 Issue 24 (30/1/03)

- Calypso 16: *CreateSprite()* function added to support **Compact Flash** sprites **(Refer to Pluto 5 Software User Manual).**

3.14 Issue 25 (5/2/03)

- Cirrus5429: Card autodetect added
- Calypso 16. Card Autodetect added

3.15 Issue 26 (7/2/03)

- Calypso 16: BUG FIX *CreateSprite()* out-by-one error when creating image mask

3.16 Issue 27 (12/2/03)

- Calypso 16: Autodetect fixes, general code cleanup.

3.17 Issue 28 (19/2/03)

- Calypso 16: Autodetect fix for sprite width problem

3.18 Issue 29 (20/3/03)

- Calypso 16: Autodetect fixes: blt_busy() timeout added, load_image_block() speeded up

3.19 Issue 30 (15/4/03)

- Calypso 16: Bug fix: crashes with VPutStr() due to uninitialised variable

3.20 Issue 31 (17/4/03)

- Calypso 16: Bug fix: double buffering not working correctly for layer BL

3.21 Issue 32 (28/4/03)

- Bug fix: ScreenFree() returning inverted value.

3.22 Issue 33 (2/6/03)

- **Major Release:** ELO and MicroTouch Touch Screen Driver added. **(Refer to Pluto 5 Software User Manual).**

3.23 Issue 34 (4/07/03)

- Calypso 16: 16 bit colour handling improved (**Note 5.2).**

3.24 Issue 35 (30/07/03)

- Calypso 16: Autodetect mode: improvements in image loading speed.

3.25 Issue 36 (08/08/03)

- Calypso 16: Bug Fix for function Line() in 16 bit colour mode

3.26 Issue 37 (05/09/03)

- Calypso 16: Bug Fix for 1ms interrupt

3.27 Issue 38 (11/09/03)

- Calypso 16: Bug Fix for double buffer flicker

3.28 Issue 39 (26/09/03)

- Calypso 16: Bug Fix for Casino Board shared interrupt handling (DUARTs + vertical)

3.29 Issue 40 (6/01/04)

- Touch Screen Driver: Fix to handle problem with some ELO touchscreens. These sometimes transmit incomplete touch packets when in streaming mode, causing error in reported screen XY co-ordinates.

3.30 Issue 41 (19/02/04)

- Security PIC Driver fixes. **(Note 5.3)**
- Stepper Driver fixes (**Note 5.4**)
- Touch Screen Driver fixes

3.31 Issue 42 (17/03/04)

- Stepper Driver fix: Function GetMotorPosn() now correctly returns number of symbols and not number of steps.

3.32 Issue 43 (05/05/04)

- Calypso 16: Timing fixes to improve Cremson to video memory bandwidth. The fix involves modifying register settings to optimise performance with ISSI 42S16400 SDRAMs. For more information please refer to:

  Pluto 5 Technical Bulletin No 12: Pluto 5/ Calypso16 Graphics Controller Limitations.

3.33 Issue 44 (07/05/04)

- Heber Internal Release – No functional changes.

3.34 Issue 45 (13/05/04)

- Cremson video driver – bug fix for 16-bit transparent sprites.

3.35 Issue 46 (09/06/04)

- Security PIC driver: bug fix for ClearPICLog and GetPICLog functions

3.36 Issue 47 (28/09/04)

- PIC driver: bug fix reading leap years

3.37 Issue 48 (20/11/04)

- Stepper driver modification: Even though the final position was ALWAYS correct with the current driver, an extra rotation was required by the stepper driver to assert this position correctly. Technical support has decided to limit the driver to only check the position of the reel on a RAMP_UP and RUN vector as RAMP_DOWN was creating problems.

3.38 Issue 49 (24/01/05)

- The rtc and e2rom drivers have been added to the standard peripheral library. The rtc and e2rom are optional devices on the Pluto5 Casino board but the drivers are now available to allow customers to use the devices if required. **(Note 5.6)**
- CDROM driver: Improvement by making it compatible with Cremson and Calypso Enhanced drivers for 16 bit colour images.

3.39 Issue 49r2 (11/01/06)

Stepper driver improvement: It has also been noticed that in particular occasion, the reels stutters during the initialization process (one customer with 200 step reels). Therefore, 2 new functions were created to “tweak” the duty cycle specific to the initialization process.
(See Pluto 5 Software User Manual for more information).

void setWakeUpDelay(UWORD setval) where setval will then be the new delay of duty cycle within initialization

UWORD getWakeUpDelay(void) which will return the current delay of duty cycle within initialization

3.40 Issue 50 (05/02/07)

There has been a problem with the stepper driver. It would not detect and error unless it passed the first tab. A new function

IsHardwareFailureDetected(const DEVICE *dev, WORD motidx) will return is there is error with the reel.

4 INTERFACE LIBRARY (81-16178)

4.1 Issue 1 (4/5/00)

- First Release

4.2 Issue 2 (7/12/01)

- Modifications to support Calypso 16

4.3 Issue 3 (9/12/02)

- FAT32 Filesystem Driver added for IDE **Compact Flash** and **Hard Drive**

4.4 Issue 4 (13/03/03)

- Modification to FAT32 Driver. To fix **Compact Flash** bug causing **VERY SLOW FILE LOADING**

4.5 Issue 5 (2/07/03)

- Correction to sprite engine to correctly handle double buffering for all layers **(Refer to Pluto 5 Software User Manual).**

4.6 Issue 6 (07/08/03)

- Correction to sprite engine function CreateSprite() in 16 bit colour mode

4.7 Issue 7 (8/09/03)

- Major improvements to Compact Flash FAT32 File system Driver **(Refer to Pluto 5 Software User Manual).**

4.8 Issue 8 (6/01/04)

- Modification to FAT32 Driver. To fix bug in identifying End Of Cluster marker, which sometimes caused the driver to **hang when listing directories**.

4.9 Issue 9 (07/05/04)

- Heber Internal Release – No functional changes.

4.10 Issue 10 (22/10/04)

- FAT32 driver: ChangeDir function implemented more efficiently to save stack space.

4.11 Issue 11 (16/11/04)

- FAT32 driver: Fix to allow access to CF cards that are supplied with the partition active flag marked as *inactive.* This flag is found in the boot sector partition table.

4.12 Issue 12 (24/01/05)

- Heber internal release.

4.13 Issue 13 (11/01/06)

- FAT32 driver: Internal improvement of the driver transparent on user side:
    - Compact flash card compatibility.
    - MountFAT32() will report more information about the card in case of problem using Fat32 file system.
    - Inter locking system implemented to avoid multiple file access (stop data loss)
    - Better error handling within driver.

Please see Pluto 5 Software User Manual for more information.

5 NOTES

5.1 Byte Ordering From Compact Flash

When loading image files in Image Alchemy .raw format, the byte order is incorrect: the high byte and low byte within a word is reversed.

Issue 23 of the Peripherals Library has a new function: **CremsonByteOrder()**
This sets the byte order when loading video memory.

Example:

```
#define NORMAL 0
#define REVERSE 1

        /* Load Image */
        CremsonByteOrder(&CremsonDevice, REVERSE);
        if (!FAT32LoadFile(&ATADevice, "a1.raw",(UWORD*)(VideoMemBase+offscreen)))
        {
                PutImage(&CremsonDevice,0,0,0,(UWORD*)(VideoMemBase+offscreen));
        }
        else
        {
                TxString(&SerialDevice,1,"\r\nFile not found.\r\n");;
        }
        CremsonByteOrder(&CremsonDevice, NORMAL);
```

5.2 16 Bit Colour Handling Improvements

Two new functions have been added to improve the handling of 16-bit colour in the Cremson Driver. The functions are:

5.2.1 SetColorMode() Set the current colour mode

void **SetColorMode**(const DEVICE* *dev*, UBYTE *mode*)

parameters:

dev:	&CremsonDevice	
mode:	**INDIRECT_COLOR**	8 bit per pixel colour depth
	DIRECT_COLOR	16 bit per pixel colour depth

5.2.2 GetColorMode() Return the current colour mode

UBYTE **GetColorMode**(const DEVICE* dev)

return value:

INDIRECT_COLOR	8 bit per pixel colour depth
DIRECT_COLOR	16 bit per pixel colour depth

5.2.3 Example

The following example shows how **SetColorMode()** is used in applications with mixed colour depths, for example, a 16 bit colour *base layer* and 8 bit colour *console* and *middle layers.*

```
void VDemo()
{
const DEVICE* dev;
LONG screen0, screen1, screen2, sprite1, sprite2, sprite3, free;

dev=&CremsonDevice;

/* Layers in memory:
screen 0 - base layer      16 bit color
screen 1 - console layer  8 bit color palette A
screen 2 - middle layer    8 bit color palette B
*/

/* memory addresses */
screen0  = 0;
screen1  = screen0 + HSIZE*VSIZE*2;   // 16 bit layer so memory x 2
screen2  = screen1 + HSIZE*VSIZE;
sprite1  = screen2 + HSIZE*VSIZE;

VideoOn(dev);
DisableLayer(dev, BASE_ENABLE | MIDDLE_ENABLE | CONSOLE_ENABLE);

/* Initialise BASE layer:    screen0, 16 bit color (no palette) */

SetColorMode(dev, DIRECT_COLOR);
InitLayer(dev, &CremsonLayer[BL_LAYER], DIRECT_COLOR, LAYER_FRAME0, BLACK, HSIZE, VSIZE,
screen0, screen0, screen0, screen0);
ClearLayer(dev, &CremsonLayer[BL_LAYER], LAYER_FRAME0, HSIZE);

/* Initialise CONSOLE layer: screen1,  8 bit color */

SetColorMode(dev, INDIRECT_COLOR);
InitLayer(dev, &CremsonLayer[C_LAYER], INDIRECT_COLOR, LAYER_FRAME0, BLACK, HSIZE, VSIZE,
screen1, screen1, screen1, screen1);
ClearLayer(dev, &CremsonLayer[C_LAYER], LAYER_FRAME0, HSIZE);
LoadCPalette(dev,paletteC_table);                          /* Load console layer palette */

/* Initialise MIDDLE layer:  screen2,  8 bit color */

SetColorMode(dev, INDIRECT_COLOR);
InitLayer(dev, &CremsonLayer[ML_LAYER], INDIRECT_COLOR, LAYER_FRAME0, BLACK, HSIZE, VSIZE,
screen2, screen2, screen2, screen2);
ClearLayer(dev, &CremsonLayer[ML_LAYER], LAYER_FRAME0, HSIZE);
LoadMBPalette(dev,paletteMB_table);                    /* Load middle/base layer palette */

/* Load background image and sprites */
LoadImage(dev, bgnd, screen0, bgnd_width*bgnd_height*2);     /* Load 16 bit color background */
sprite2 =
LoadSprite(dev, cherry, sprite1, cherry_width*cherry_height*2);  /* Load 16 bit color cherry sprite */
sprite3 =
LoadSprite(dev, coin, sprite2, coin_width*coin_height);            /* Load 8 bit color coin sprite */
free    =
LoadSprite(dev, queen, sprite3, queen_width*queen_height);   /* Load 8 bit color queen sprite */
EnableLayer(dev, BASE_ENABLE| MIDDLE_ENABLE | CONSOLE_ENABLE);


/* 16 bit color BLIT: cherry sprite to the Base Layer, screen 0 at x=200, y=200 */
SetColorMode(dev, DIRECT_COLOR);
PutSprite(dev, screen0,   200, 200, sprite1, cherry_width, cherry_height, 1);

/* 8 bit color BLIT:  coin sprite to the Console Layer, screen 1  at x=200, y=0 */
SetColorMode(dev, INDIRECT_COLOR);
PutSprite(dev, screen1, 200, 0, sprite2, coin_width, coin_height, 1);

/* 8 bit color BLIT queen sprite to the Middle Layer, screen 2 at x=400, y=200*/
PutSprite(dev, screen2, 400, 200, sprite3, queen_width, queen_height, 1);

while(1);

}
```

5.3 PIC Driver Fixes

The functions provided for reading data from the security PIC sometimes worked unreliably.
This has been greatly improved by modifying the timing of the I2C communication with the PIC device.
It is advised that all PIC access should still have a single retry in case of failure.

It should be noted that communication with the PIC is a very slow, due to the low clock speed of the PIC. We strongly advise not to retrieve alarms from the PIC in an interrupt or a video animation loop as it will probably cause unacceptable execution delays (screen freezing, interrupt overrun).

Example:

```
if(!GetPICLog(&PICDevice, -5 ,&power_off_reading))
        GetPICLog(&PICDevice, -5 ,&power_off_reading);

/*
read the 5th alarm log:
NOTE the minus sign is mandatory
*/
```

A log entry is a LONG variable defined as follow:

```
typedef union
        {
        ULONGinteger;
        struct
                {
                BYTE            swstatus;
                BYTE            high;
                BYTE            mid;
                BYTE            low;
                } log;
        } PICLOG;
```

swstatus is the status of the switches at the time of the alarm.

Integer & 0x00ffffff; is a number of minutes from 01/01/2000 0h0 to the minute your alarm triggered.
This number of minutes needs to be used to calculate when was the alarm triggered.

5.4 Reel Driver Fixes

GetStepMotorTabsMissed() now works correctly. This function returns the number of times a reel was not seen when expected by the opto sensor.

This variable is only cleared by *ClearStepMotorErrors().* Previously it was cleared each rotation when the position was checked.

Other fixes were implemented but they should be transparent to the game programmer:

- The accuracy in tracking the reel position has been improved. The position is now also tracked during *ramp-up* and *ramp-down* as well as during *run*
- The position of the reels is now corrected for all final positions. Previously, the position was not corrected if the final position was just before the tab.
- Function *SpinStepMotor()* now works correctly for all values of parameter *distance*. Previously some values caused a non-return condition leaving the reel spinning forever.

5.5 DUART bug fix

As the devkit was released, in some occasions, customers are reporting flickers on the lamps. Heber technical support advises to follow the following step to fix the flicker:

5.5.1 Decrease the number of wait states to access RAM or EPROM faster:

By modifying FPGA.h in the project directory from :

```
#define CSAM0 0x00FFFFFD
#define CSAM1   0x00FFFFFD
```

to

```
#define CSAM0 0x00FFFFF5
#define CSAM1   0x00FFFFF5
```

Making these changes in fpga.h will reduce the time to access data from ram (or ROM[1]) and decrease the amount of time spent in the 1ms interrupt in charge of updating reels or lamps. The 1ms interrupt is almost the highest priority and it is very important to spend as less time as possible in it.

5.5.2 Raising the priority of the 1ms interrupt

In FPGA.c, in the sim40 configuration, the periodic interrupt control register |PICR| is set as below

```
const SIM40CFG sim40_cfg=
{
....
  0x0440,     /* PICR */
.....
};
```

This gives a higher priority to the duart list (video display vertical interrupt + DUARTs) than the 1ms interrupt which controls most of the device update. Making the change describe below, will raise the 1ms interrupt priority above the duart list priority.

```
const SIM40CFG sim40_cfg=
  {
....
  0x0640,     /* PICR */
.....
  };
```

Usually, at this point customers report that the flicker has disappeared.

[1] Be aware that this change will not work if the board is working with a single EPROM (8 bit data bus)

5.5.3 Side effects

The above changes in interrupt priority can cause the malfunctioning of the duart drivers. A further fix in the Hardware Library has fixed this.

This fix has been implemented in version 36 of the Hardware Library. For further explanation on how it has been fixed, please email: support@heber.co.uk.

5.6 RTC and E2ROM driver

A full description of these two drivers is available in the Pluto 5 Software User Manual version 6 and later.

These drivers have been previously available on request to customers. They have now been included in the libraries. This may cause dependency problems in certain circumstances. Please contact support@heber.co.uk for further assistance.

In the Real time clock driver source code which is invisible for the user, two variables are used to track the time:

```
extern WORD rtc_year;
extern BYTE year_reg;
```

These variables have been declared extern for a programmer to choose if he wants the time to be backed up on power down or not. In order for the time to be backed up on a power down, these two variables need to be declared either in idram.c or sysram.c as follow:

```
WORD  rtc_year;
BYTE   year_reg;
```

6 UPDATING PLUTO 5 LIBRARY

After downloading the latest library, copy the content of the zip file in:[2]

C:\heber\libs\hardware\pluto5\lib\81-16082
Or
C:\heber\libs\interface\pluto5\lib\81-16178
Or
C:\heber\libs\peripherals\pluto5\lib\81-16083

Update you project makefile to target the correct libraries as follow (where XX is the current issue of the library):

```
lhdw=   ../../libs/hardware/$(target)/lib/81-16082/81-16082-XX
lper=   ../../libs/peripherals/$(target)/lib/81-16083/81-16083- XX
lint=   ../../libs/interface/$(target)/lib/81-16178/81-16178- XX
ihdw=   ../../libs/hardware/$(target)/lib/81-16082/81-16082- XX
iper=   ../../libs/peripherals/$(target)/lib/81-16083/81-16083- XX
iint=   ../../libs/interface/$(target)/lib/81-16178/81-16178- XX
```

Ensure the linker scripts (lnkram.x and lnkrom.x) are targeting the latest hardware library:

```
vectors_start    = .;
../../libs/hardware/pluto5/lib/81-16082/81-16082-XX/vectors.o(.text)
vectors_end      = .;
```

Open a DOS prompt where you project lives and type the following commands:

“touch *c” to update all the dependencies
“make” to remake you current project using the latest libraries.

[2] The following explanation assumes the Development Kit was installed in C:\heber\