Pippin Video

version 002

Abstract: This document contains `PippinVideo.h` and `myPippinVideo.c`. `PippinVideo.h` is a header file that contains enumerated constants and structures necessary to access functionality unique to the Pippin video architecture. `myPippinVideo.c` is sample code that illustrates features unique to the Pippin video architecture.

Please send questions and comments via e-mail to pippindev@apple.com.

1 PippinVideo.h

```
// ==========================================================================================
//
//    PippinVideo.h
//
//    Copyright 1995, Apple Computer
//
//    send comments/bugs: pippindev@apple.com
//
//    (change this to a real header)
//
//    This file contains enumerated constants and structures necessary
//    to access functionality unique to the Pippin video architecture
//
// ==========================================================================================

enum {
  kSetProgressiveScan = 0,// Progressive Scan
  kSetInterlace, // Interlaced scan, no convolution
  kSetInterlaceConv,// Interlaced scan, Convolution, no scaling
  kSetInterlaceConvScale// Interlaced scan, convolution, and scaling
};

#definecscSetScanMode128// Used to select a scan mode for the Taos driver
#definecscGetScanMode128

// structure for Requesting/Controlling Scan Mode information
typedef struct VDControlMode VDControlMode;
struct VDControlMode
{
  UInt32 csModeSelect;
  Ptr    csBaseAddressA;
  Ptr    csBaseAddressB;
};
```

2 myPippinVideo.c

```
//============================================================================
//
//    myPippinVideo.c
//
//    send comments/bugs pippindev@apple.com
//
//    v1.1 March 28, 1996
//
//    (change this to a real header)
//
//    Sample code illustrating features unique to the Pippin video architecture
//    Shows how to use driver calls to change video scan modes
//
//============================================================================


#include <stdio.h>
#include <stdlib.h>
#include <Memory.h>
#include <Displays.h>
#include <Video.h>
#include "PippinVideo.h"

// Prototypes
unsigned long GetUserInputData (void);
void          DoCurrentSettings(void);
void          DoSetScanMode(void);

// Globals
OSErr         err;
short         currentDepth;
GDHandle      myGDHandle;
short         videoDriverRefNum;
short         currentScanMode;
DisplayIDType currentDisplayType;
UInt32        currentBaseAddress = 0;
short         currentPage;

//============================================================================
//
//    main()
//
//    Starting point for everything
//
//============================================================================


void main(void)
{
  OSErr              err;
  VDControlModemyControlModeStruct;
  VDDisplayConnectInfoRecmyVDDisplayConnectInfoRec;
  VDSwitchInfoRecswitchInfo;
  CntrlParam     pBlock;
  short              input = 1;

  MaxApplZone();
```

```
  // ••• Need to check for Pippin Gestalt Selecter here

  printf("\nThis demo illustrates special functionality of the Pippin video
      architecture\n");
  printf("\n\tInsert explanation and other stuff here\n\n");

  DoCurrentSettings();
  while ( input != 999 )
  {
      printf("\nPlease enter a selection:\n");
      printf("  1 = Set Scan Mode\n");
//    printf("  2 = Swap Video Pages\n");
      printf("  3 = Current Settings\n");
      printf("input (999 to exit)>");
      input = GetUserInputData();
      switch ( input )
      {
          case 1:
              DoSetScanMode();
              break;
          case 2:
//            DoSwapPages();
              break;
          case 3:
              DoCurrentSettings();
              break;
          case 999:
              break;
          default:
              printf("Invalid selection\n");
              break;
      }
  }
  printf("\nPress Command-Q to Quit . . .\n");
}

//============================================================================
//
//    DoCurrentSettings()
//
//    Gets Current Video Stuff by calling PBStatus repeatedly
//        cscGetScanMode
//        cscGetConnection
//        cscGetMode <-- with different pages
//
//============================================================================

voidDoCurrentSettings(void)
{
  OSErr                     err;
  VDControlMode             myControlModeStruct;
  VDDisplayConnectInfoRec   myVDDisplayConnectInfoRec;
  VDPageInfo                myVDPageInfoRec;
  VDSwitchInfoRec           switchInfo;
  CntrlParam                pBlock;

  printf("\n");
```

```
printf("Scanning for driverRefNum: ");
// We get the First video device because we know there is only one device hooked
// up to Pippin
myGDHandle = DMGetFirstScreenDevice(dmOnlyActiveDisplays);
videoDriverRefNum = (*myGDHandle)->gdRefNum;
printf("%i\n", videoDriverRefNum);

printf("Current Scan Mode: ");
pBlock.ioCompletion = nil;
pBlock.ioCRefNum = videoDriverRefNum;
pBlock.csCode = cscGetScanMode;
*(Ptr *)&pBlock.csParam[0] = (Ptr)&myControlModeStruct;
err = PBStatus( (ParmBlkPtr)&pBlock, true );
if ( err )
    printf("Unsupported Functionality, err = %i\n", err);
else
{
    currentScanMode = myControlModeStruct.csModeSelect;
    switch (myControlModeStruct.csModeSelect)
    {
        case kSetProgressiveScan:
            printf("kSetProgressiveScan\n");
            break;
        case kSetInterlace:
            printf("kSetInterlace\n");
            break;
        case kSetInterlaceConv:
            printf("kSetInterlaceConv\n");
            break;
        case kSetInterlaceConvScale:
            printf("kSetInterlaceConvScale\n");
            break;
        default:
            printf("Err -- unknown Scan Mode, %i\n",
                myControlModeStruct.csModeSelect );
    }
}

printf("Current Display Type: ");
pBlock.ioCRefNum = videoDriverRefNum;
pBlock.csCode = cscGetConnection;
*(Ptr *)&pBlock.csParam[0] = (Ptr)&myVDDisplayConnectInfoRec;
err = PBStatus( (ParmBlkPtr)&pBlock, true );
currentDisplayType = myVDDisplayConnectInfoRec.csDisplayType;

switch ( currentDisplayType )
{
    case kVGAConnect:
        printf("kVGAConnect\n");
        break;
    case kNTSCConnect:
        printf("kNTSCConnect\n");
        break;
    case kPALConnect:
        printf("kPALConnect\n");
        break;
    default:
        printf(" -- something else (not VGA, NTSC, or PAL)\n");
        break;
```

```
  }

  printf("Current Screen Depth: ");
  currentDepth = (*((*myGDHandle)->gdPMap))->pixelSize;
  printf("%i\n", currentDepth);
  printf("Page Info:\n");
  printf("  Current Page: ");
  pBlock.csCode = cscGetMode;
  *(Ptr *)&pBlock.csParam[0] = (Ptr)&myVDPageInfoRec;
  err = PBStatus( (ParmBlkPtr)&pBlock, true );
  currentPage = myVDPageInfoRec.csPage;
  printf("%i\n", currentPage);
  pBlock.csCode  = cscGetBaseAddr;
  myVDPageInfoRec.csPage = 0;
  *(Ptr *)&pBlock.csParam[0]= (Ptr)&myVDPageInfoRec;
  err = PBStatus( (ParmBlkPtr)&pBlock, true );
  printf("  Page: %i, Base Address: %X\n", myVDPageInfoRec.csPage,
      myVDPageInfoRec.csBaseAddr);

  myVDPageInfoRec.csPage = 1;
  err = PBStatus( (ParmBlkPtr)&pBlock, true );
  printf("  Page: %i, Base Address: %X\n", myVDPageInfoRec.csPage,
      myVDPageInfoRec.csBaseAddr);
  printf("  Page Count: ");
  pBlock.csCode  = cscGetPages;
  err = PBStatus( (ParmBlkPtr)&pBlock, true );
  printf("%i\n", myVDPageInfoRec.csPage);

  printf("\n");
}

//============================================================================
//
//    DoSetScanMode()
//
//    Request and set the scan mode;
//
//============================================================================


voidDoSetScanMode(void)
{
  short      input = 1;
  CntrlParampBlock;
  VDControlModemyControlModeStruct;

  pBlock.ioCompletion = nil;
  pBlock.ioCRefNum = videoDriverRefNum;
  pBlock.csCode = cscSetScanMode;
  *(Ptr *)&pBlock.csParam[0] = (Ptr)&myControlModeStruct;

  while ( input != 999 )
  {
      printf("\nPlease enter a selection:\n");
      printf("  1 = Progressive Scan\n");
      printf("  2 = Interlace\n");
      printf("  3 = Interlace, Convolution\n");
      printf("  4 = Interlace, Convolution, Scaling\n");
      printf("input (999 to exit)>");
```

```
        input = GetUserInputData();

        if ( input == 999 ) break;

        if ( input == 1 )
        {
            if ( currentDisplayType != kVGAConnect ) printf("err -- not in VGA
                mode\n");
            else
            {
                myControlModeStruct.csModeSelect = kSetProgressiveScan;
                err = PBControl( (ParmBlkPtr)&pBlock, true );
            }
        }
        else
        {
            if ( currentDisplayType == kVGAConnect ) printf("err -- not in NTSC/PAL
                mode\n");
            else
            {
                if (input == 2) myControlModeStruct.csModeSelect = kSetInterlace;
                else if (input == 3) myControlModeStruct.csModeSelect =
                    kSetInterlaceConv;
                else if (input == 4) myControlModeStruct.csModeSelect =
                    kSetInterlaceConvScale;
                err = PBControl( (ParmBlkPtr)&pBlock, true );
            }
        }
        if (err) printf("err in PBControl: %i", err);
    }
}

//============================================================================
//
//    GetUserInputData()
//
//    Whats until the user inputs anything other than 0
//
//============================================================================


unsigned long GetUserInputData (void)
{
    longtempLong = 0;
    do scanf ("%ld", &tempLong); while (tempLong == 0);
    return (tempLong);
}
```