# Intel Z3735F POX DLL Dynamic Bank Instruction（C#）

## Initialize Bank

[DllImport("AclasDevices.dll")]

public static extern void AclasDevice\_Initialize();//initialize bank

public static extern void AclasDevice\_ReadDevice(int nDeviceCMD, ASDSOnDataEvent OnDataEvent); //rfid、ibutton Function

public static extern void AclasDevice\_CustomerDisplayASCII(int nDevice, IntPtr lpData, int Len); // customer display characters display function

public static extern void AclasDevice\_CustomerDisplayCMD(Int32 nCMD); // customer display cls/backlight function

public static extern void AclasDevice\_CustomerDisplayImage(int nDevice,int nWidth,int nHeight, IntPtr lpData); //image display function

public static extern bool AclasDevice\_OpenDrawer(int nDevicenOpenDrawer); // drawer function

## RFID（only has card reading function temporarily and other functions will be added later）

nDeviceCMD: devRFID // select read type

const int devRFID = 0x05; //RFID

public delegate void ASDSOnDataEvent(IntPtr Data, Int32 nSize); // callback

AclasDevice\_ReadDevice(devRFID,new ASDSOnDataEvent(OnData));

private void OnData(IntPtr Data, Int32 nSize) // callback

{

switch (Marshal.ReadByte((IntPtr)Data)) //// first byte of data

{

case 0x31: //// rfid data

}

}

## IBUTTON

nDeviceCMD: devIButton // select read type

const int devIButton = 0x06; //Ibutton

public delegate void ASDSOnDataEvent(IntPtr Data, Int32 nSize); // callback

AclasDevice\_ReadDevice(devIButton,new ASDSOnDataEvent(OnData));

private void OnData(IntPtr Data, Int32 nSize) // callback

{

switch (Marshal.ReadByte((IntPtr)Data)) //// first byte of data

{

case 0x1a: /// ibuttion data

}

}

## Drawer

nDevicenOpenDrawer parameter 1、2

// eg. Open drawer 1

AclasDevice\_OpenDrawer(1);

// return false failure true success

## Customer Display

// Device ID（every device ID stands for one customer display, choose the customer display you need when do development）

nDevice parameter

constint devCD6 = 0x600; //9 bit VFD, only support ASCII mode, 9 digits

constint devCD7 = 0x700; //2\*20（5\*7）bitmap, only support ASCII mode, 2 lines, 20 characters each line（5\*7）

constint devCD8 = 0x800; //24\*128 bitmap, only support image mode and ASCII mode, when on ASCII mode, single-character 12\*6, able to display 2 lines, at most 21 characters each line.

constint devCD9 = 0x900; //LCD 32\*144, support image mode and ASCII mode, when on ASCII mode, single-character 16\*8, able to display 2 lines, at most 18 characters each line.

constint devCD10 = 0xa00; //LCD 65\*132, support image mode and ASCII mode, when on ASCII mode, single-character 16\*8, able to display 4 lines, at most 16 characters each line.

constint devCD11 = 0xb00; //LCD 32\*240, support image mode and ASCII mode, when on ASCII mode, single-character 16\*8, able to display 2 lines, at most 30 characters each line.

constint devCD13 = 0xd00; //6characters + 8segment code display, only support ASCIImode, 6 characters + 8digits.

**ASCII Mode（only for US-ASCII）**

// eg 32\*144 display

nDevice: devCD9 // parameter

const int devCD9 = 0x900; //LCD 32\*144

byte[] bytes = Encoding.Default.GetBytes("123"); // encode all the characters of the specified string into a byte sequence

// IntPtr lpData = System.Runtime.InteropServices.Marshal.AllocHGlobal(bytes.Length); // memory allocation

// System.Runtime.InteropServices.Marshal.Copy(bytes, 0, lpData, bytes.Length); // copy data into memory

// AclasDevice\_CustomerDisplayASCII(devCD9, lpData, bytes.Length); display 123

**Image Mode（fit for non-US-ASCII）**

private static List<byte> GetBitmapData(System.Drawing.Bitmap bitmap) //// convert image into bitmap data function

{

try

{

int bitmapWidth = bitmap.Width;

int bitmapHeight = bitmap.Height;

int cols = bitmapWidth / 8;

List<byte> data = new List<byte>();

for (int rowIndex = 0; rowIndex < bitmapHeight; rowIndex++)

{

for (int colIndex = 0; colIndex < cols; colIndex++)

{

int startColIndex = colIndex \* 8;

byte pixelByte = 0;

for (int index = 0; index <= 7; index++)

{

if ((index + startColIndex) < bitmapWidth)

{

if (bitmap.GetPixel(index + startColIndex, rowIndex).ToArgb() == System.Drawing.Color.Black.ToArgb())

{

pixelByte += (byte)(Math.Pow(2, (7 - index)));

}

}

}

data.Add(pixelByte);

}

}

return data;

}

catch (Exception ex)

{

return null;

}

}

// eg. display image a.bmp（a.bmp generated by the upper computer, when use 32\*144 display, the image size will be 32\*144）

nDevice: devCD9 // parameter

const int devCD9 = 0x900; // select customer display LCD 32\*144

System.Drawing.Bitmap bitmap = new System.Drawing.Bitmap("a.bmp"); //// image

List<byte> resultList = GetBitmapData(bitmap); // call image into bitmapdata function

if (resultList != null)

{

IntPtr lpReponseData = ystem.Runtime.InteropServices.Marshal.AllocHGlobal(resultList.Count); // memory allocation

System.Runtime.InteropServices.Marshal.Copy(resultList.ToArray(),0, lpReponseData, resultList.Count); // copy data into memory

AclasDevice\_CustomerDisplayImage(devCD9, bitmap.Width, bitmap.Height, lpReponseData); /// display image, data sending indicator

**}**

**Clear Screen/Backlight Command**

// constant definitions

// nCMD parameter

// const int cmdCD6Clear = 0x603; // clear 9 bit VFD

// const int cmdCD7Clear = 0x703; // clear 2\*20（5\*7）bitmap

// const int cmdCD8Clear = 0x803; // clear 24\*128 bitmap

const int cmdCD9Off = 0x901; // backlight off LCD 32\*144

const int cmdCD9On = 0x902; // backlight on LCD 32\*144

const int cmdCD9Clear = 0x903; // clear LCD 32\*144

const int cmdCD10Off = 0xa01; // backlight off LCD 65\*132

const int cmdCD10On = 0xa02; // backlight on LCD 65\*132

const int cmdCD10Clear = 0xa03; // clear LCD 65\*132

const int cmdCD11Off = 0xb01; // backlight off LCD 32\*240

const int cmdCD11On = 0xb02; // backlight on LCD 32\*240

const int cmdCD11Clear = 0xb03; // clear LCD 32\*240

const int cmdCD13Off = 0xd01; // backlight off 6 characters + 8 segment code

const int cmdCD13On = 0xd02; // backlight on 6 characters + 8 segment code

const int cmdCD13Clear = 0xd03; //clear 6 characters + 8 segment code

//

// eg. 32\*144 clear screen

nCMD：cmdCD9Clear; /// 32\*144 CLS

AclasDevice\_CustomerDisplayCMD(cmdCD9Clear);

## Unloading Bank

[DllImport("AclasDevices.dll")]

public static extern void AclasDevice\_Finalize();