The I2C way in UEFI reads the TP's id.

The old rule, first say to achieve the function: use I2C way to read the TP ID, and then through the TP ID different to do LCD compatibility.

File path:BOOT.XF.1.4/boot\_images/QcomPkg/Sdm660Pkg/Library/MDPPlatformLib/MDPPlatformLib.c.

1, define the pin foot to be used.

#define CDP\_TP\_INT\_GPIO 67 //TP中断pin脚

#define CDP\_TP\_RES\_GPIO 66 //TP重置pin脚

#define CDP\_TP\_I2C\_SDA\_GPIO 14 //i2c数据读写pin脚

#define CDP\_TP\_I2C\_CLK\_GPIO 15 //i2c时钟控制pin脚

2, the introduction of relevant header file, define global variables.

/\* I2C Interfaces \*/

#include <Protocol/EFII2C.h>

static i2c\_config cfg;

static void \*pI2cHandle = NULL;

3, i2c's init function.

i2c\_status i2c\_init(UINT32 SlaveAddr, UINT32 I2cFreq)

{

i2c\_status i2cstatus = I2C\_SUCCESS;

cfg.bus\_frequency\_khz = I2cFreq;

cfg.slave\_address = SlaveAddr;

cfg.slave\_address\_type = I2C\_07\_BIT\_SLAVE\_ADDRESS;

MDP\_Status Status = MDP\_STATUS\_OK;

EFI\_TLMM\_PROTOCOL \*TLMMProtocol = NULL;

if (EFI\_SUCCESS != gBS->LocateProtocol(&gEfiTLMMProtocolGuid, NULL, (void \*\*)&TLMMProtocol))

{

DEBUG((EFI\_D\_ERROR, "DisplayDxe: Locate TLMM protocol failed!\n"));

Status = MDP\_STATUS\_NO\_RESOURCES;

}

else

{

/\* Setup tp i2c sda Pin \*/

if (EFI\_SUCCESS != TLMMProtocol->ConfigGpio((UINT32)EFI\_GPIO\_CFG(CDP\_TP\_I2C\_SDA\_GPIO, 2, GPIO\_OUTPUT, GPIO\_PULL\_DOWN, GPIO\_2MA), TLMM\_GPIO\_ENABLE))

{

DEBUG((EFI\_D\_WARN, "DisplayDxe: Configure GPIO %d for Reset\_N line Failed!\n", CDP\_TP\_I2C\_SDA\_GPIO));

}

/\* Setup tp i2c clk Pin \*/

if (EFI\_SUCCESS != TLMMProtocol->ConfigGpio((UINT32)EFI\_GPIO\_CFG(CDP\_TP\_I2C\_CLK\_GPIO, 2, GPIO\_OUTPUT, GPIO\_PULL\_DOWN, GPIO\_2MA), TLMM\_GPIO\_ENABLE))

{

DEBUG((EFI\_D\_WARN, "DisplayDxe: Configure GPIO %d for Reset\_N line Failed!\n", CDP\_TP\_I2C\_CLK\_GPIO));

}

/\* Setup tp int Pin \*/

if (EFI\_SUCCESS != TLMMProtocol->ConfigGpio((UINT32)EFI\_GPIO\_CFG(CDP\_TP\_INT\_GPIO, 0, GPIO\_OUTPUT, GPIO\_PULL\_UP, GPIO\_2MA), TLMM\_GPIO\_ENABLE))

{

DEBUG((EFI\_D\_WARN, "DisplayDxe: Configure GPIO %d for Reset\_N line Failed!\n", CDP\_TP\_INT\_GPIO));

}

/\* Setup tp reset Pin \*/

if (EFI\_SUCCESS != TLMMProtocol->ConfigGpio((UINT32)EFI\_GPIO\_CFG(CDP\_TP\_RES\_GPIO, 0, GPIO\_OUTPUT, GPIO\_PULL\_UP, GPIO\_2MA), TLMM\_GPIO\_ENABLE))

{

DEBUG((EFI\_D\_WARN, "DisplayDxe: Configure GPIO %d for Reset\_N line Failed!\n", CDP\_TP\_RES\_GPIO));

}

/\* Set tp int line HIGH \*/

if (EFI\_SUCCESS != TLMMProtocol->GpioOut((UINT32)EFI\_GPIO\_CFG(CDP\_TP\_INT\_GPIO, 0, GPIO\_OUTPUT, GPIO\_NO\_PULL, GPIO\_2MA), GPIO\_HIGH\_VALUE))

{

DEBUG((EFI\_D\_WARN, "DisplayDxe: Reset\_N line HIGH failed!\n"));

}

MDP\_OSAL\_DELAYUS(150);

/\* Set tp reset line HIGH \*/

if (EFI\_SUCCESS != TLMMProtocol->GpioOut((UINT32)EFI\_GPIO\_CFG(CDP\_TP\_RES\_GPIO, 0, GPIO\_OUTPUT, GPIO\_NO\_PULL, GPIO\_2MA), GPIO\_HIGH\_VALUE))

{

DEBUG((EFI\_D\_WARN, "DisplayDxe: Reset\_N line HIGH failed!\n"));

}

MDP\_OSAL\_DELAYMS(8);

/\* Set tp int line LOW \*/

if (EFI\_SUCCESS != TLMMProtocol->GpioOut((UINT32)EFI\_GPIO\_CFG(CDP\_TP\_INT\_GPIO, 0, GPIO\_OUTPUT, GPIO\_NO\_PULL, GPIO\_2MA), GPIO\_LOW\_VALUE))

{

DEBUG((EFI\_D\_WARN, "DisplayDxe: Reset\_N line HIGH failed!\n"));

}

MDP\_OSAL\_DELAYMS(60);

}

i2cstatus = i2c\_open((i2c\_instance) (I2C\_INSTANCE\_004), &pI2cHandle);

if (I2C\_SUCCESS != i2cstatus)

{

DEBUG((EFI\_D\_ERROR, "Failed to initialize I2C %d\n", i2cstatus));

}

return i2cstatus;

}

4, i2c read data.

unsigned int i2c\_read\_reg(unsigned int addr)

{

uint32 bRead = 0;

unsigned int getdata = 0;

i2c\_status i2cstatus = I2C\_SUCCESS;

unsigned char rdbuf[2] = {0};

gBS->Stall(600000);

i2cstatus = i2c\_read (pI2cHandle, &cfg, addr, 2, rdbuf, 1, &bRead, 2500);

if(I2C\_SUCCESS != i2cstatus)

{

DEBUG((EFI\_D\_ERROR, "Read addr:0x%X error\n", addr));

}

gBS->Stall(600000);

getdata=rdbuf[0] & 0x00ff;

getdata<<= 8;

getdata |=rdbuf[1];

DEBUG((EFI\_D\_ERROR, "[dong]rdbuf[0] & 0x00ff is %d\n\n", rdbuf[0] & 0x00ff));

return (rdbuf[0] & 0x00ff);//getdata;//

}

5, i2c write data.

unsigned int i2c\_write\_reg(unsigned char addr, unsigned int reg\_data)

{

uint32 bWrote = 0;

i2c\_status i2cstatus = I2C\_SUCCESS;

unsigned char wdbuf[2] = {0};

wdbuf[1] = (unsigned char)(reg\_data & 0x00ff);

wdbuf[0] = (unsigned char)((reg\_data & 0xff00)>>8);

i2cstatus = i2c\_write (pI2cHandle, &cfg, addr, 1, wdbuf, 2, &bWrote, 2500);

if(I2C\_SUCCESS != i2cstatus)

{

DEBUG((EFI\_D\_ERROR, "Write addr:0x%X data:0x%X error\n", addr, reg\_data));

}

return bWrote;

}

6, i2c off.

i2c\_status i2c\_deinit()

{

return i2c\_close(pI2cHandle);

}

7, i2c read tp's id.

unsigned int i2c\_read\_tp\_sensor(void)

{

unsigned int tp\_sensor\_id = 0;

i2c\_init(0x14,400000);//i2c从机地址、读写速率

tp\_sensor\_id = i2c\_read\_reg(0x814a);//tp中存id的寄存器地址

i2c\_deinit();

return tp\_sensor\_id;

}

————————————————