

XCAM M480 SDK

|  |  |
| --- | --- |
| Version: | 1.2 |
| Date: | 22th March 2018 |
| Author: | LAU Yat Fei Jackson |

|  |  |  |
| --- | --- | --- |
| **Revision** | **Date** | **Comment** |
| 1.0 | 6 Dec 2017 | Original release |
| 1.1 | 30 Jan 2018 | Multiple Color Palette + POIs |
| 1.2 | 22 March 2018 | Adaptive colour palette |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

**Contents**

[XCAM FOUNDATION 3](#_Toc509315971)

[**Project Setting** 4](#_Toc509315972)

[**Header** 4](#_Toc509315973)

[**Software dependencies** 4](#_Toc509315974)

[**Preprocessor Symbols** 4](#_Toc509315975)

[**ThermalSensorAPI\_M480.lib** 5](#_Toc509315976)

[**XCAM Constant** 5](#_Toc509315977)

[**Customised data structure** 6](#_Toc509315978)

[**API** 9](#_Toc509315979)

[Sensor Connection 14](#_Toc509315980)

[Build and Test 14](#_Toc509315981)

**XCAM FOUNDATION**

The documentation of the XCAM SDK for users.

The XCAM SDK basically contains fundamental functions which supports for MCU **M480** series library. Below is the whole project structure of XCAM\_M480.

m480-bsp-realchip\_org

|---- Library

|---- CMSIS

|---- Device/Nuvoton/M480

|---- Include

|---- M480.h

|---- system\_M480.h

|---- Source

|---- system\_M480.c

|---- ARM/startup\_M480.s

|---- StdDriver/src

|---- clk.c

|---- retarget.c

|---- sys.c

|---- i2c.c

|---- timer.c

|---- uart.c

|---- husbd.c

|---- ThermalAPI\_inC

|---- KEIL/obj/ThermalSensorAPI\_M480.lib

|---- MI\_XCAM.h

|---- ThermalSensor.h

|---- M480.h

|---- SampleCode

|---- StdDriver/USBD\_UVC\_HTPA32\_R1\_32x32\_UART

|---- KEIL/MI\_XCAM\_M480.uvproj

|---- KEIL/Nu\_Link\_Driver.ini

|---- KEIL/obj

|---- MI\_XCAM\_M480.bin

|---- MI\_XCAM\_M480.axf

|---- descriptors.c

|---- HUART.c

|---- main.c

|---- Table\_UVC.c

|---- thermal\_i2c.c

|---- thermal\_i2c.h

|---- usbd\_video.h

|---- usbd\_video.c

**SDK Version: Release V3.01 180322**

## **Project Setting**

### **Header**

#include "ThermalSensor.h"

#include “MI\_XCAM.h”

#include "M480.h"

#include "usbd\_video.h"

### **Software dependencies**

..\..\..\..\Library\CMSIS\Include

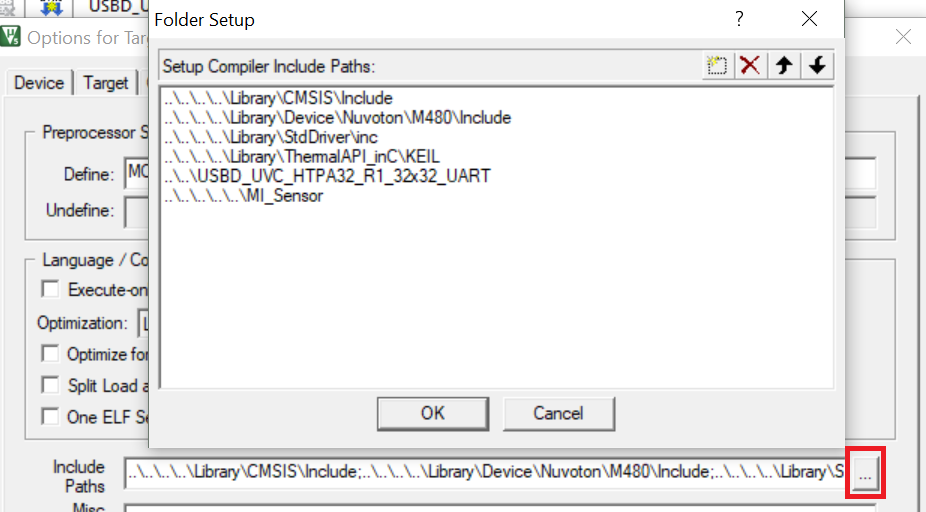
..\..\..\..\Library\Device\Nuvoton\M480\Include

..\..\..\..\Library\StdDriver\inc

..\..\..\..\Library\ThermalAPI\_inC\KEIL

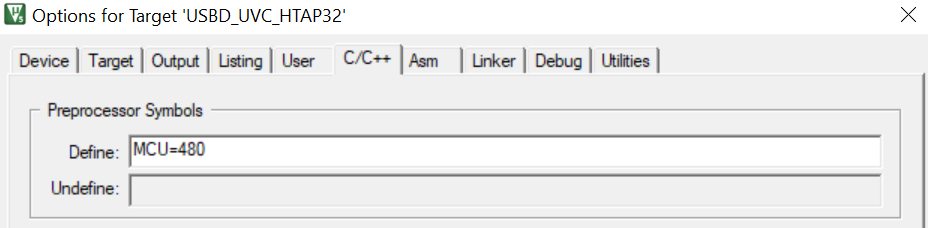
..\..\USBD\_UVC\_HTPA32\_R1\_32x32\_UART

..\..\..\..\..\MI\_Sensor

****

### **Preprocessor Symbols**

Define: MCU=480



### **ThermalSensorAPI\_M480.lib**

#define DEADPIXELCOMPENSATE

Usage: Dead pixel(s) compensation using average masking.

Prerequisite: EEPROM should have the information of

1) Number of dead pixel and

2) Coordinates of dead pixels

\*If there are any dead pixels next to each other, sensor should be rejected since we are using averaging masking

\*Pre-set from ThermalSensorAPI\_M480.lib

#define POI

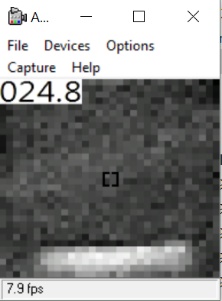
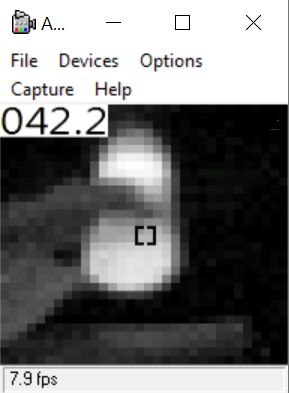
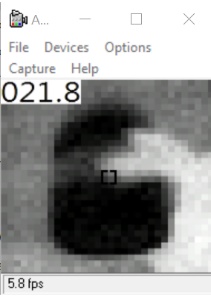
Usage: Point of interests (POIs) recording for each frame. If not define, then FRAMEPOIS object will be empty.

\*Pre-set from ThermalSensorAPI\_M480.lib

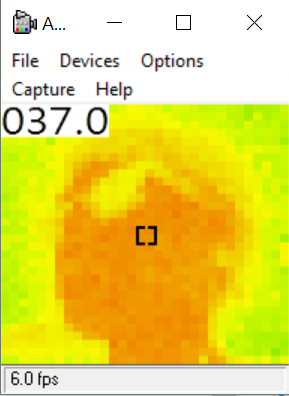
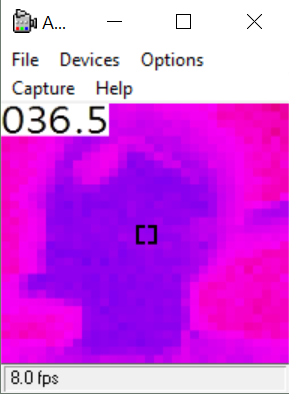
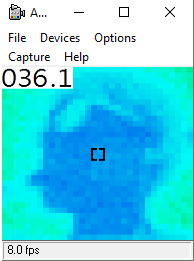
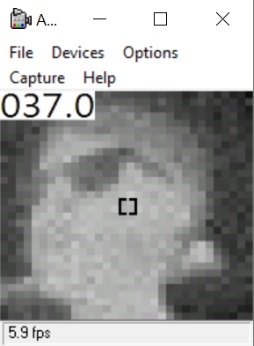
#define COLOR\_ADAPTIVE

Usage: The FRAMEPOIS determine how the temperature spread on colour palette

Better to use monotonic RGB colour table under adaptive mode, i.e.: COLORPALETTE\_BW\_ADAPTIVE

#define COLORPALETTE0/1/2/3/\_BW\_STEP10/\_BW\_STEP25

Usage: Define colour palette mode

### **XCAM Constant**

|  |  |  |
| --- | --- | --- |
| **Variables** | **Values** | **Usage** |
| WIDTH | 32 | Width of the sensor |
| HEIGHT | 32 | Height of the sensor |
| W\_WIDTH | 2 | Width of temperature calculation window |
| W\_HEIGHT | 2 | Height of temperature calculation window |

### 

### **Customised data structure**

|  |
| --- |
| peri\_interface : Peripheral interface |
| enum peri\_interface{  HUART = 0,  UART,  SPI }; |

|  |
| --- |
| YUV\_COLOR\_INFO\_T: YUV data structure |
| typedef struct YUV\_COLOR\_INFO\_T {  unsigned int YUVData;  } YUV\_COLOR\_INFO\_T; |

|  |
| --- |
| RGB\_COLOR\_INFO\_T: RGB data structure |
| typedef struct RGB\_COLOR\_INFO\_T{  unsigned char R;  unsigned char G;  unsigned char B;  } RGB\_COLOR\_INFO\_T; |

|  |
| --- |
| REGISTERSETTING : Data structure storing the values of sensor register. |
| typedef struct REGISTERSETTING{  unsigned short MBIT;  unsigned short BIAS;  unsigned short CLOCK;  unsigned short BPA;  unsigned short PU;  } REGISTERSETTING; |

|  |
| --- |
| EEPROM: Data structure storing values read from EEPROM. |
| typedef struct EEPROM{  float PixCMin; // Minimum sensitivity coefficient, used for scaling  float PixCMax; // Maximum sensitivity coefficient, used for scaling  unsigned short gradScale; // Emissivity factor  unsigned short TableNumberSensor; // The look-up table number of sensor belongs  unsigned short epsilon; // Factor for fine tuning of the sensitivity for all Pixel  REGISTERSETTING calibRegister; // Sensor register values for calibration  unsigned short VddRef; // used supply voltage during calibration measured  float PTATGrad; // Factor of calculating ambient temperature (Ta)  float PTATOff; // Factor of calculating ambient temperature (Ta)  unsigned char VddScaling; // VddComp scaling coefficient  unsigned short VddScalingOff; // VddComp scaling coefficient  unsigned char GlobalOffset; // Factor for fine tuning of the sensitivity for all Pixel  unsigned short GlobalGain; // Factor for fine tuning of the sensitivity for all Pixel  REGISTERSETTING userRegister; // Sensor register values for user  unsigned short DevID; // Device ID  unsigned char NrOfDefPix; // Number of dead pixel(s)  unsigned short DeadPixAdr[MAXNROFDEFECTS]; // Array of dead pixel addr  unsigned char DeadPixMask[MAXNROFDEFECTS]; // Array of dead pixel mask  signed short VddGrad[ELAMOUNT]; // VddComp gradient  signed short VddOff[ELAMOUNT]; // VddComp offset  signed short ThGradN[Pixel]; // thermal gradient  signed short ThOffN[Pixel]; // compensate for any thermal drifts  unsigned long PixCN[Pixel]; // Sensitivity coefficients  } EEPROM; |

|  |
| --- |
| SENSORSETTING: Data structure storing sensor look-up table values for TO calculation. |
| typedef struct SENSORSETTING{  signed long TABLENUMBER;  signed long long PCSCALEVAL; // defined scaling coefficient  signed long NROFTAELEMENTS;  signed long NROFADELEMENTS;  signed long TAEQUIDISTANCE;  signed long ADEQUIDISTANCE;  signed long ADEXPBITS;  signed long TABLEOFFSET;  unsigned char MBITTRIMDefault;  signed long SensRv;  unsigned int\* TempTable;  unsigned int\* XTATemps;  unsigned int\* YADValues;  } SENSORSETTING; |

|  |
| --- |
| TEMPIXEL: POI data structure |
| typedef struct TEMPIXEL{  unsigned short x;  unsigned short y;  signed short Tmp;  } TEMPIXEL; |

|  |
| --- |
| FRAMEPOI: Storing POIs of single frame |
| typedef struct FRAMEPOI{  TEMPIXEL maxTemPixel;  TEMPIXEL maxTemPixel;  } FRAMEPOIS; |

### **API**

**ThermalSensor.h**

void M480\_InitSensor(void);

void M480\_OpenSensor(void);

void M480\_StartSensor(void);

**MI\_XCAM.h**

void InitI2C(unsigned char mode);

void HighDensSequentialRead(unsigned short address,unsigned char\* data, unsigned short numbytes);

void HighDensPageWrite(unsigned short address,unsigned char\* data, unsigned short numbytes);

void ReadCalibDataN(void);

void InitSensorDev(unsigned short TN);

void InitMBITTRIMN(unsigned char user);

void InitBIASTRIMN(unsigned char user);

void InitBPATRIMN(unsigned char user);

void InitPUTRIMN(unsigned char user);

void InitCLKTRIMN(unsigned char user);

void GetImageData (void);

unsigned short GetTemp(unsigned int x, unsigned int y);

FRAMEPOIS GetFramePOIs (void);

unsigned short GetTempDisplay(void);

int GetTargetPixelIndex(void);

unsigned int CalcTO(unsigned int TAmb, signed int dig, signed long PiC, unsigned int dontCalcTA);

unsigned int StartStreaming(int Mode, char Temps, char Stream);

void SetTempDisplay(unsigned short flag);

void SetTargetPixelIndex(int index);

void ResetFramePOIs(void);

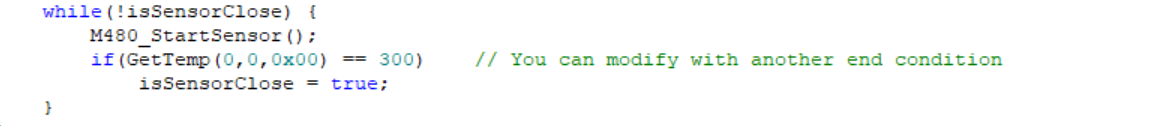
|  |  |
| --- | --- |
| void | M480\_InitSensor(void); |

Open and setup the system environment for I2C.

|  |  |
| --- | --- |
| void | M480\_OpenSensor(void); |

Initialization of sensor, mainly read calibration data and setup trim registers.

|  |  |
| --- | --- |
| void | M480\_StartSensor(void); |

XCAM start streaming.

|  |  |
| --- | --- |
| void | InitI2C(unsigned char mode); |

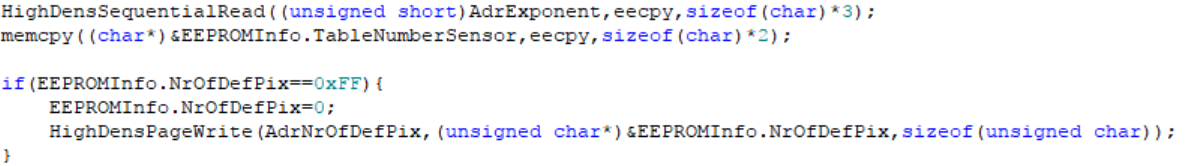


Setup the EEPROM & Sensor address to the device.

Dependencies: 0🡪 Init I²C for Sensor (> 1000 kHz)

1🡪 Init I²C for EEPROM (max 400 kHz)

|  |  |
| --- | --- |
| void | HighDensSequentialRead(unsigned short address,unsigned char\* data, unsigned short numbytes); |
| void | HighDensPageWrite(unsigned short address,unsigned char \*data, unsigned short numbytes); |



HighDensSequentialRead() & HighDensPageWrite() reads/writes multiple bytes from a high-density (>= 32 Kb) serial EEPROM device.

Dependencies: 'address' contains address word

'data' contains the reading result

'numbytes' contains the length of bytes to read

|  |  |
| --- | --- |
| void | ReadCalibDataN(void); |



Read all required calibration data from sensor EEPROM, details in TYPEDEF structure EEPROM.

|  |  |
| --- | --- |
| void | InitSensorDev (unsigned short TN); |



Open and setup the sensor device, pick up the setting corresponding to the type of optics by its TableNumber.

|  |  |
| --- | --- |
| void | InitMBITTRIMN(unsigned char user); |



Dependencies: 0🡪 Setting during Calibration

1🡪 Setting from user

Range: 4 <= m <= 12

Initialization of Trim Register 1 MBIT/ (m+4) bit as ADC resolution to sensor register.

|  |  |
| --- | --- |
| void | InitBIASTRIMN(unsigned char user); |



Dependencies: 0🡪 Setting during Calibration

1🡪 Setting from user

Range: 0 to 31 🡺 1μA to 13μA

Initialization of Trim Register 2, adjust the bias current of the ADC.

|  |  |
| --- | --- |
| void | InitCLKTRIMN(unsigned char user); |



Dependencies: 0🡪 Setting during Calibration

1🡪 Setting from user

Range: 0 to 63 🡺 1MHz to 13MHz

Initialization of Trim Register 4, clock frequency setting CLK\_TRM.

|  |  |
| --- | --- |
| void | InitBPATRIMN(unsigned char user); |



Dependencies: 0🡪 Setting during Calibration

1🡪 Setting from user

Range: 0 to 31 🡺 0.2μA to 4.0μA

Initialization of Trim Register 5, adjust the common mode voltage of the preamplifier.

|  |  |
| --- | --- |
| void | InitPUTRIMN(unsigned char user); |



Dependencies: 0🡪 Setting during Calibration

1🡪 Setting from user

Range: “1000” = 100 kOhm; “0100” = 50 kOhm; “0010” = 10 kOhm; “0001” = 1 kOhm

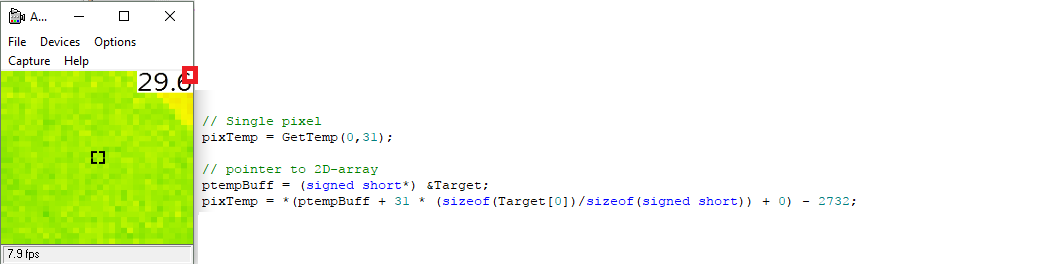
Initialization of Trim Register 7, select internal pull up resistor on SDA/SCL.

|  |  |
| --- | --- |
| Void | GetImageData(void); |



GetImageData() calculate voltage values of each pixels.

|  |  |
| --- | --- |
| unsigned short | GetTemp(unsigned int x, unsigned int y); |



GetTemp(x,y) gets the calculated temperature from 2D-array sensor, in single pixel.

\* P.S: If you want to get the whole 2D-array, you may use

ptempBuff = (signed short\*) &Target;

pixTemp = \*(ptempBuff + x \* (sizeof(Target[0])/sizeof(signed short)) + y) - 2732;

Dependencies: unsigned int x = x-coordinate of target pixel (0 < x < COLUMN-1)

unsigned int y = y-coordinate of target pixel (0 < y < ROW-1)

Return Code: Caelcius °C x 10 (e.g: 301 = 30.1°C)

998 Coordinate input ERROR

|  |  |
| --- | --- |
| FRAMEPOIS | GetFramePOIs(void); |

GetFramePOIs return the FRAMEPOIS object.

Return: data structure FRAMEPOIS

|  |  |
| --- | --- |
| void | ResetFramePOIs(void); |

ResetFramePOIs() reset the FRAMEPOIS object.

|  |  |
| --- | --- |
| unsigned int | CalcTO(unsigned int TAmb, signed int dig, signed long PiC, unsigned int dontCalcTA); |



CalcTO() calculate the object temperature via look-up table.

Dependencies: TAmb = ambient temperature

dig = pixel voltage

PiC = pixel sensitivity coefficients

Return: Object Temperature in dK

|  |  |
| --- | --- |
| unsigned int | StartStreaming(int Mode, char Temps, char Stream); |



StartStreaming() starts the streaming of the sensor, main sequence can be seen in the interrupt.

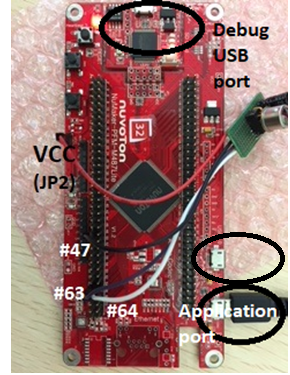
Dependencies: Mode = Initialize(1) or Normal streaming (0)

Temps: whether it is calculating temperature

Stream = whether it is streaming mode

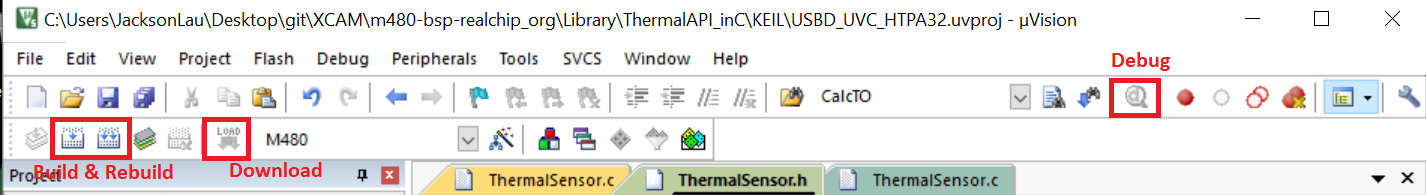
Return: 0x0 / 0xFF

**Sensor Connection**



* Please connect the USB2.0 instead of 1.1

**Build and Test**



1. Build main project successfully
2. Download binary file(.bin) to MCU

Connect the USB debug port to PC, then click "Download".

1. Application:

Disconnect the debug USB port and connect the USB to application port, then open AMCAP/any camera application.

Debug:

Keep connecting the debug USB port and press “Debug” to get into debug mode.