

XCAM N329- SDK

|  |  |
| --- | --- |
| Version: | 1.0 |
| Date: | 5th Dec 2017 |
| Author: | LAU Yat Fei Jackson |

|  |  |  |
| --- | --- | --- |
| Revision | Date | Comment |
| 1.0 | 5 Dec 2017 | Original release |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

**Contents**

[XCAM Foundation 3](#_Toc500320004)

[Project Setting 4](#_Toc500320005)

[Header 4](#_Toc500320006)

[Software dependencies 4](#_Toc500320007)

[Preprocessor Symbols 6](#_Toc500320008)

[ThermalSensorAPI\_N329.lib 6](#_Toc500320009)

[Constant Variables 6](#_Toc500320010)

[Customised data structure 7](#_Toc500320011)

[API 9](#_Toc500320013)

[Build and Test 14](#_Toc500320014)

[UART/HUART Interface 14](#_Toc500320015)

[Setting 14](#_Toc500320016)

[Output commands 15](#_Toc500320017)

**XCAM Foundation**

The documentation of the XCAM SDK with user code.

The XCAM SDK basically contains functions provided by sample code, which supports for MCU **N329-** series library. Below is the whole project structure of XCAM\_N329.

|----N32901-3\_Non-OS\_KEIL

|----I2C

|----lib/w55fa93\_i2c.h

|----HTPA32\_0619\_R1\_XCam

|----example/SwLib3

|---- I2CS/DrvI2C.c

|---- I2CS/DrvI2C.h

|---- demo.h

|---- HUART.c

|---- main.c

|---- MI\_sensor.h

|---- standalone.c

|---- Table\_UVC.c

|---- usbd\_video.c

|--- -wb\_init.s

|---- **KEIL # User**

|----MI\_Thermal.uvproj

|----MI\_Thermal/MI\_XCAM\_N329\_SW\_I2C.bin

|----lib

|----videoclass\_HTPA32.h

|----w55fa93\_UVC\_YUB\_Only.lib

|----N32901

|----N32901.sct

|----SYSLIB/Lib

|----w55fa93\_reg.h

|----W55FA93\_syslib.lib

|----wberrcode.h

|----wbio.h

|----wblib.h

|----wbtypes

|----ThermalAPI\_inC

|----lib/videoclass\_HTPA32.h

|----example/demo/KEIL/**ThermalSensor.h**

|----example/demo/KEIL/**MI\_XCAM.h**

|----example/demo/KEIL/demo\_Data/N32901\_SingleTask/**ThermalSensorAPI\_N329.lib #SDK**

|----UDC

|----lib/usbd.h

|----lib/w55fa93\_UDC.lib

|----VPOST

|----lib/w55fa93\_vpost.h

|----lib/W55FA93\_VPOST.lib

**SDK Version: Release V2.01 171208**

## **Project Setting**

### **Header**

"ThermalSensor.h”

“w55fa93\_vpost.h”

“videoclass\_HTPA32.h”

“demo.h”

### **Software dependencies**

..\..\..\lib

..\..\..\..\SYSLIB\Lib

..\..\..\..\I2C\lib

..\..\..\..\UDC\lib

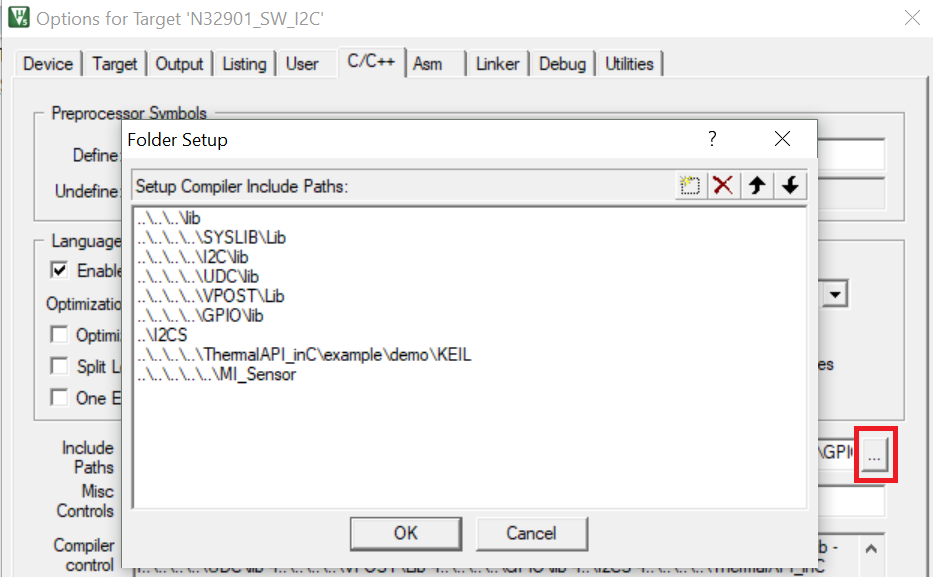
..\..\..\..\VPOST\Lib

..\..\..\..\GPIO\lib

..\I2CS

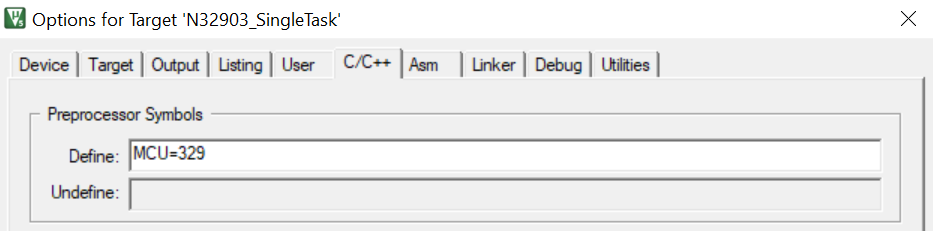
..\..\..\..\ThermalAPI\_inC\example\demo\KEIL

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

****

### **Preprocessor Symbols**

Define: MCU=329



### **ThermalSensorAPI\_N329.lib**

#### **Constant Variables**

|  |  |  |
| --- | --- | --- |
| **Variables** | **Values** | **Usage** |
| Pixel | 1024 | Number of total pixels (32x32=1024) |
| PixelEighth | 128 |  |
| ROW/HEIGHT | 32 | Number of row in sensor |
| COLUMN/WIDTH | 32 | Number of column in sensor |
| W\_WIDTH | 2 | Width of middle window for average temperature |
| W\_HEIGHT | 2 | Height of middle window for average temperature |
| PTATamount | 8 |  |
| ELOFFSET | 1024 | Start address of el. Offset |
| ELAMOUNT | 256 |  |
| ELAMOUNTHALF | 128 |  |
| StackSize | 16 |  |
| PTATSTARTADSRESS | 1282 |  |
| VDDADDRESS | 1280 |  |
| GetElEveryFrameX | 10 | amount of normal frames to capture after which the el. Offset is fetched |
| STACKSIZEPTAT | 30 |  |
| STACKSIZEVDD | 50 |  |
| VddStackAmount | 30 |  |
| MAXNROFDEFECTS | 24 | Maximum number of defect pixels |
| TEMP\_OFFSET | 4.5 | Temperature offset |

#### **Customised data structure**

|  |
| --- |
| 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; |

#### **API**

**ThermalSensor.h**

void N329\_InitSensor(void);

void N329\_OpenSensor(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);

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 | InitI2C(unsigned char mode); |

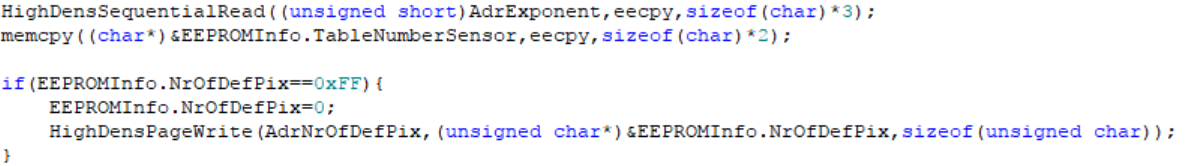


Setup the EEPROM & Sensor address to the device.

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.

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



Dependencies: 0🡪 Setting during Calibration

1🡪 Setting from user

Range: 0 to 63 🡺 1MHz to 13MHz

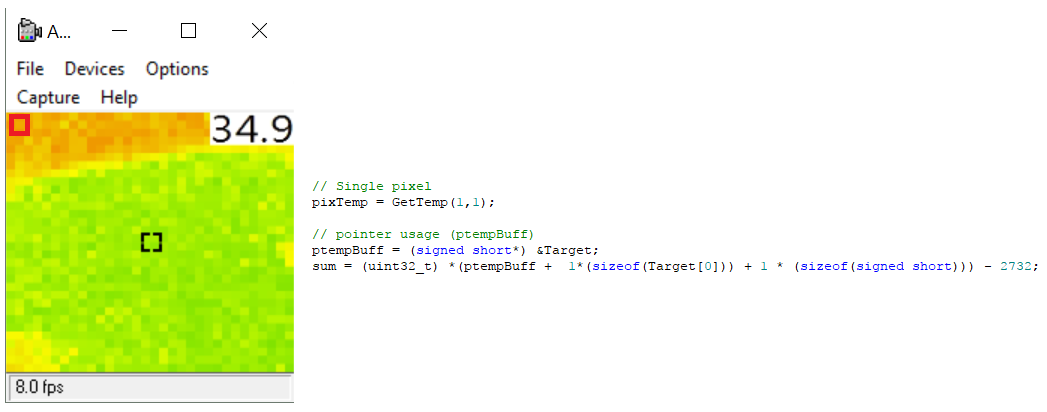
Initialization of Trim Register 4, clock frequency setting CLK\_TRM.

|  |  |
| --- | --- |
| Void | 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])) + y \* (sizeof(signed short))) - 2732;

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

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

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

998 Coordinate input ERROR

|  |  |
| --- | --- |
| 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

|  |  |
| --- | --- |
| void | N329\_InitSensor(void); |



Open and setup the system environment for I2C.

|  |  |
| --- | --- |
| void | N329\_Open(void); |

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

**Build and Test**

1. Build main project successfully
2. Download binary file(.bin) to MCU
3. Run the camera application (e.g.: AMCAP)

**UART/HUART Interface**

## Setting

1. Main.c: main() 🡪 sysUartPort(input)

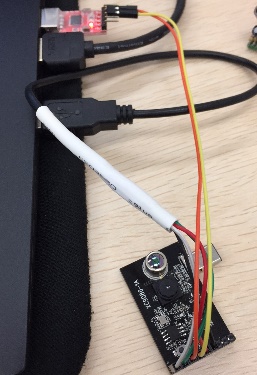
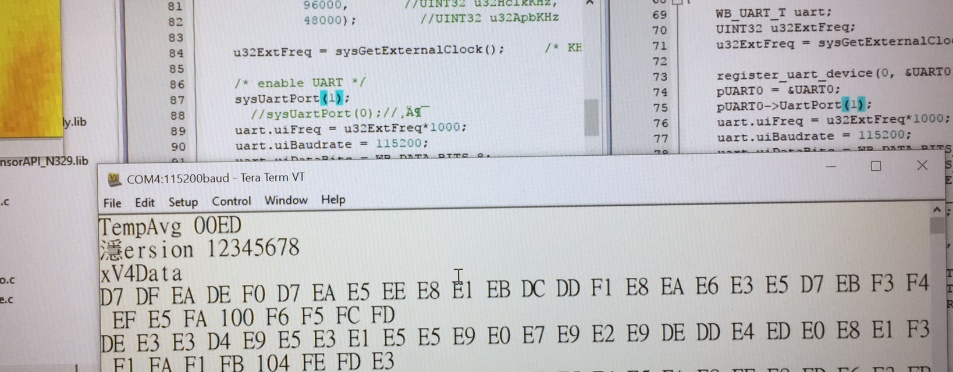
0 (HUART) , 1 (UART)

1. HUART.c: HUART\_init() 🡪 pUART0->UartPort(input)

Uart0(HUART), 1(Debug data using UART)

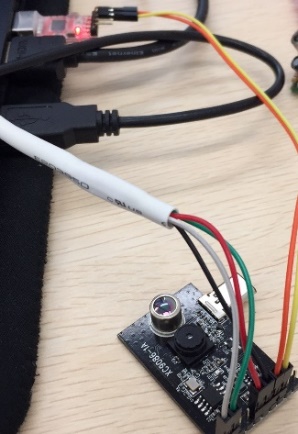
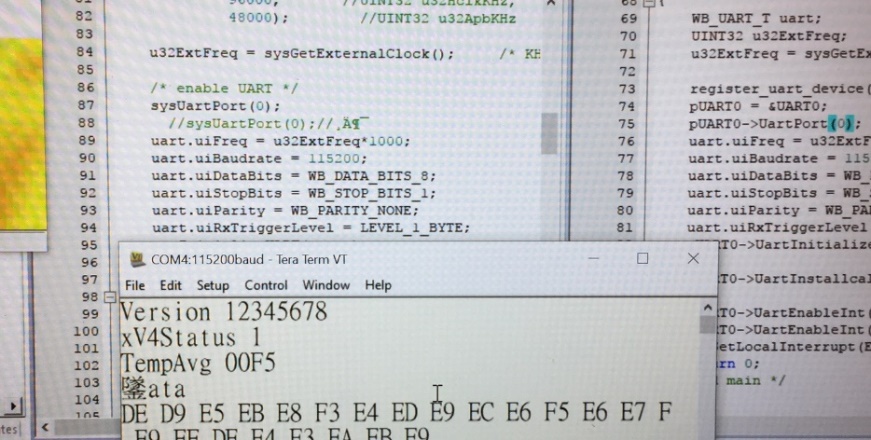
UART

sysUartPort(1) + pUART0->UartPort(1)



HUART

sysUartPort(0) + pUART0->UartPort(0)



## Output commands

Please also refers to UartDataValid\_Handler() in **HUART.C**

VCMD=$COMMAND

|  |  |  |
| --- | --- | --- |
| AVG(Average Temperature) | DAT(Temperature Data/Array) | RES(Resolution) |
| STA(Status) | VER(Version) |  |