MT3620 M4

User Manual

Version: 0.1
Release date: November 1, 2019

Document Revision History

Revision	Date	Description
0.1	1 November 2019	Initial release.

M

Table of Contents

Document Revision History **2**
Table of Contents **3**
1. Introduction **4**
1.1 System overview 4
1.1.1 MT3620 Cortex M4 Main Features 4
1.1.2 MT3620 Block Diagram 5
1.2 Azure Sphere Software Development 5
1.2.1 M4 code – How to 6
1.3 Available Resources 8
1.3.1 Available Resources from Microsoft 8
1.3.2 Available Resources from Mediatek 8
1.3.3 Available Resources from AVNET 8
2. M4 Software Support **9**
2.1 RTOS integration 9
2.2 SDK - CMSIS integration 9
2.2.1 CMSIS-Core 9
2.2.2 CMSIS-Driver 9
2.2.3 CMSIS-DSP 9
2.2.4 CMSIS-NN 9
2.3 MT3620 M4 M-HAL API 9
2.3.1 M-HAL API document 10
2.3.2 M-HAL API Code 10
2.3.3 Sample code 10
2.4 Inter-Core Communication 10
2.5 MT3620 M4 Power Management 11
2.5.1 M4 Power Mode 11
2.5.2 M4 Power Saving Control Scheme 11
2.6 Memory Mapping 11
3. General **13**

1. Introduction

This document is to describe the MT3620 M4 software programming methodology and shows the available materials. Which could be divided into five chapters. The first chapter comes with the MT3620 M4 hardware capability and provides the M4 software programming how to. The second chapter describes some enhanced topics including operation system, M4 API reference manual, and power management. And the final chapter is the general usage reminding.

1.1 System overview

MT3620 features two general purpose ARM Cortex-M4F I/O subsystems, each of which runs at up to 200MHz. These subsystems were designed to support real-time requirements when interfacing with a variety of on-chip peripherals including UART, I2C, SPI, I2S, and ADC. They are completely general-purpose Cortex-M4F units which may be tailored to specific application requirements. On-chip peripherals may be mapped to any of the three end-user accessible cores, including the CA7.

MT3620 also includes over 5MB of embedded RAM, split among the various cores. There is a fully-integrated PMU and a real-time clock. Flash memory is integrated in the MT3620 package. Please refer to the “Azure Sphere MT3620 Support Status” document from Microsoft for information about how much memory and which hardware features are available to end-user applications. Only hardware features supported by the Azure Sphere system are available to MT3620 end-users.

1.1.1 MT3620 Cortex M4 Main Features

- Two general purpose ARM Cortex M4 cores, each with 192kB TCM, 64kB SRAM, and integrated FPU; ideal for real-time control requirements
- In-package serial flash
- User-accessible cores support execute-in-place (XIP) from flash
- Five “ISU” serial interface blocks which can be configured as I2C master. I2C slave, SPI master, SPI slave, or UART; I2C runs at up to 1MHz, SPI at up to 40MHz, and UARTs at up to 3Mbps
- Two I2S interfaces supporting slave and TDM slave modes
- Eight-channel, 12-bit, 2MS/s single-ended successive approximation ADC using internal 2.5V or external 1.8V reference
- 76 programmable GPIO pins with programmable drive strength (some multiplexed with other functions)
- 12 PWM outputs
- 24 external interrupt inputs
- Six hardware counter blocks which can count and measure pulses and perform quadrature decoding
- RTC can run from dedicated 32 kHz external input, from on-die 32 kHz oscillator, from on-die ring oscillator, or from main oscillator
- One-time programmable e-fuse block for storing chip-specific information
- Two additional, dedicated UARTs, one for each CM4F I/O subsystem
- Per-core watchdog timers
- Per-core general-purpose timers

1.1.2 MT3620 Block Diagram

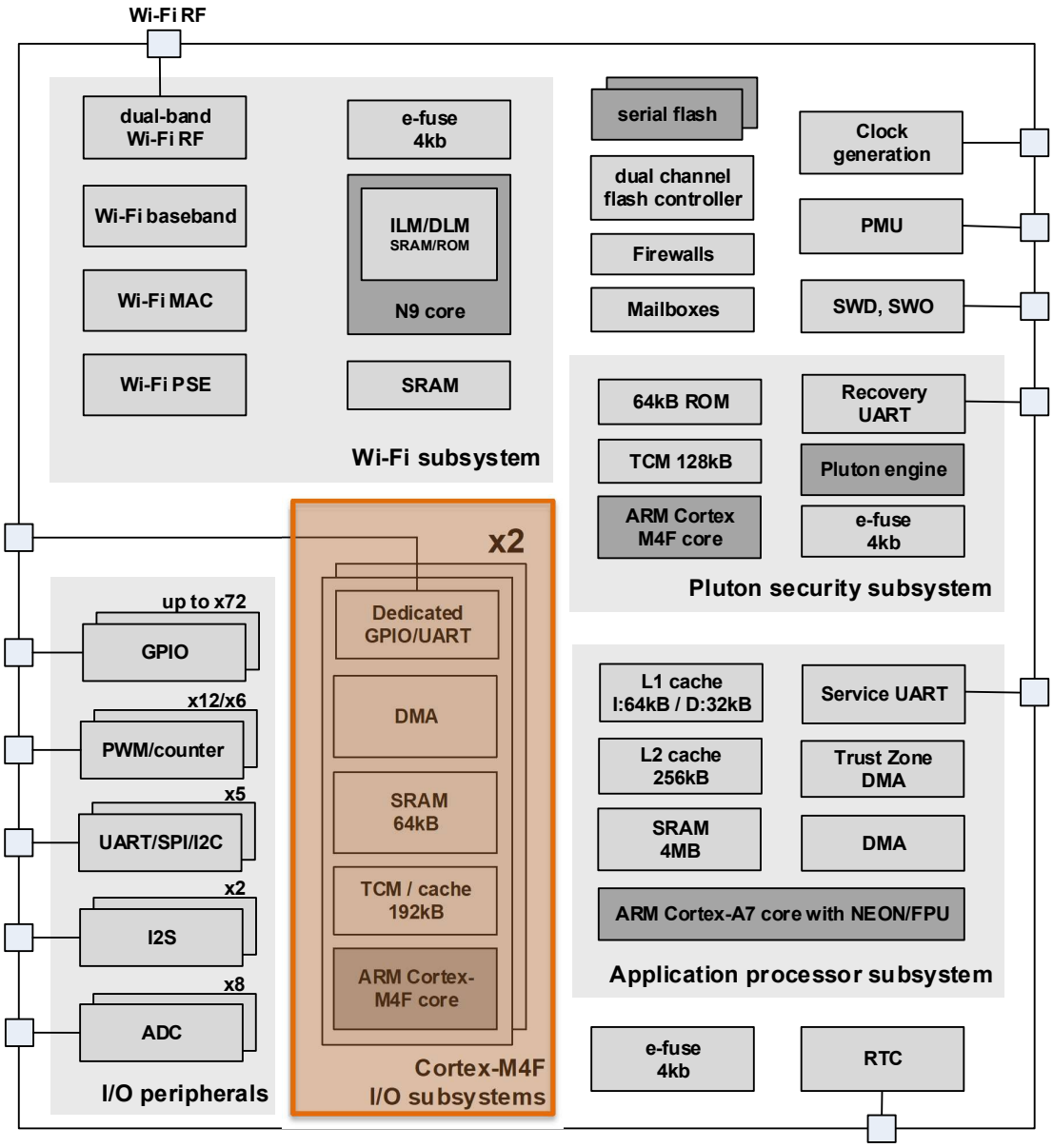

Figure 1. MT3620 Block Diagram

1.2 Azure Sphere Software Development

Please refer to the Microsoft Azure Sphere Document Website (https://docs.microsoft.com/en-ca/azure-sphere/) for the detailed information.
Here are some frequently used links

- Environment Setup / System Overview
  - Install Azure Sphere.
    - https://docs.microsoft.com/en-ca/azure-sphere/install/overview
  - Quick starts for Azure Sphere software development.

        - https://docs.microsoft.com/en-ca/azure-sphere/quickstarts/qs-overview
- SW Development
    - Azure Sphere applications overview.
        - https://docs.microsoft.com/en-ca/azure-sphere/app-development/applications-overview
    - Develop High-Level applications.
        - https://docs.microsoft.com/en-ca/azure-sphere/app-development/high-level-app-development-overview
    - Develop Real-Time capable applications.
        - https://docs.microsoft.com/en-ca/azure-sphere/app-development/rt-app-development-overview
    - Azure Sphere application libraries.
        - https://docs.microsoft.com/en-ca/azure-sphere/reference/applibs-reference/api-overview
    - Samples and reference solutions.
        - https://docs.microsoft.com/en-ca/azure-sphere/resources/sample-links
- SW Deployment
    - Load and unload application.
        - https://docs.microsoft.com/en-ca/azure-sphere/app-development/sideload-app
    - OTA (Over the Air) application deployment.
        - https://docs.microsoft.com/en-ca/azure-sphere/deployment/deployment-overview
- HW Design & Manufacturing
    - Overview
        - https://docs.microsoft.com/en-ca/azure-sphere/hardware/hardware-manufacturing-overview
    - MT3620 development board user guide
        - https://docs.microsoft.com/en-ca/azure-sphere/hardware/hardware-manufacturing-overview
    - RF (Radio Frequency) tools
        - https://docs.microsoft.com/en-ca/azure-sphere/hardware/rf-tools
- MISC Information
    - Connect to Wi-Fi
        - https://docs.microsoft.com/en-ca/azure-sphere/network/wifi-including-ble
    - Connect to Ethernet
        - https://docs.microsoft.com/en-ca/azure-sphere/network/connect-ethernet

1.2.1 M4 code – How to

Here are some “How To” to show you how MT3620 M4 software programming works.

1.2.1.1 How to build M4 application code – With Visual Studio (Quick & Fast)

Visual Studio & Azure Sphere SDK should be installed before starting to build the M4 code. Please refer to the following URL for the detailed information.

- https://docs.microsoft.com/en-us/azure-sphere/install/overview

The existing samples and reference solutions are available on GitHub. Please refer to the following URL for the detailed information.

- https://docs.microsoft.com/en-us/azure-sphere/resources/sample-links

You can use Visual Studio to develop and debug real-time capable applications (RTApps) in much the same way as high-level applications. The primary difference is that RTApps projects currently must use CMake, instead of vcxproj files. Please refer to the following URL for the detailed information.

- https://docs.microsoft.com/en-us/azure-sphere/app-development/rtapp-manual-build

1.2.1.2 How to build M4 application code – Without Visual Studio

Users could also build M4 code using Azure Sphere Developer Command Prompt (without Visual Studio). Please refer to the following URL for the detailed information.

- https://docs.microsoft.com/en-us/azure-sphere/app-development/rtapp-manual-build

1.2.1.3 How to load the M4 application binary to the flash of development board

There are two approaches to load the M4 application binary into MT3620 Flash.

1. Local side-load for testing
   - https://docs.microsoft.com/en-us/azure-sphere/app-development/sideload-app
2. OTA (Over the Air) for upgrade SW binaries at end user side.
   - https://docs.microsoft.com/en-us/azure-sphere/deployment/deployment-overview

1.2.1.4 How to manage memory

There are three types of memory available on the real-time cores.

- TCM
  - Each real-time core has 192 KB of tightly-coupled memory.
- XIP Flash
  - Execute-in-place (XIP) flash memory is shared with high-level applications. A window into the XIP mapping of the flash is visible to each core at address 0x10000000.
- SYSRAM
  - Each real-time core also has its own 64 KB of SYSRAM.

Please refer to the following URL for more detailed information.

- https://docs.microsoft.com/en-us/azure-sphere/app-development/memory-latency

1.2.1.5 How to execute M4 application binary from TCM

The M4 application could be executed in TCM or SYSRAM or XIP Flash. User could modify the linker.ld to configure the target address.
Please refer to the following URL for more detailed information.

- https://docs.microsoft.com/en-us/azure-sphere/app-development/memory-latency

1.2.1.6 How M4 application binary been loaded and how M4 boot up

No matter the M4 application binary is configured to be executed on TCM or SYSRAM or XIP Flash, it would be stored on the Flash permanently. When system boot up, the MT3620 A7 is responsible for loading the M4 application binary to the TCM or SYSRAM or don't load if execute on flash, and then set the M4 programming counter. Finally, A7 will release the M4 hardware reset signal which allows M4 to boot up.

1.2.1.7 How to debug M4 application code

There are several approaches to debug M4 application. The most straight forward way is to use the Visual Studio IDE, which allows users to set software break point and dump memory status. For users who prefer debugging with UART log, they have to connect MT3620 UART pins to computer and write their own application to print log via UART.

1.3 Available Resources

1.3.1 Available Resources from Microsoft

- Azure Sphere Official Website.
  - https://azure.microsoft.com/en-us/services/azure-sphere/
- Azure Sphere Documents.
  - https://docs.microsoft.com/en-ca/azure-sphere/

1.3.2 Available Resources from Mediatek

- MT3620 Official Website.
  - https://www.mediatek.com/products/azureSphere/mt3620
- MT3620 Product Brief.
  - https://d8602zu8ugzlg.cloudfront.net/mediatek-craft/documents/mt3620/MediaTek-MT3620-Product-Brief-PDFMT3620PBA4-0919.pdf
- MT3620 Datasheet.
  - https://d8602zu8ugzlg.cloudfront.net/mediatek-craft/documents/mt3620/MT3620-Datasheet-v1.1.pdf
- MT3620 M4 API Reference Manual: TBD
  - https://www.mediatek.com/products/azureSphere/mt3620

1.3.3 Available Resources from AVNET

- AVNET for Microsoft
  - https://www.avnet.com/shop/us/m/microsoft/
- Azure Sphere Starter Kit
  - https://www.element14.com/community/community/designcenter/azure-sphere-starter-kits/

2. M4 Software Support

2.1 RTOS integration

(To be available)

2.2 SDK - CMSIS integration

2.2.1 CMSIS-Core

(To be available)

2.2.2 CMSIS-Driver

Please refer to chapter 2.3, instead of CMSIS-Driver, MediaTek provides MT3620 M4 M-HAL API for controlling MT3620 peripherals. MediaTek does not provide the support of CMSIS-Driver.

2.2.3 CMSIS-DSP

CMSIS-DSP is a software library, which provided by ARM.

2.2.4 CMSIS-NN

CMSIS-NN is a software library, which provided by ARM.

2.3 MT3620 M4 M-HAL API

MediaTek provided the MT3620 M4 driver API which is named as “M-HAL” (Mediatek Hardware Abstraction Layer). The target of M-HAL is to wrap all IO peripheral access programming sequence into an easy to use driver API to simplify the hardware access control for developers. Users can use these API directly to control the external peripherals. There are totally 12 submodules in the M-HAL API, which includes:

- ADC
- DMA
- EINT
- GPIO
- GPIOIF
- I2C
- PWM
- SPIM
- UART
- WDT

Each submodule could be divided into 2 parts: M-HAL and HDL.

- HDL (Hardware Driving Layer) - Handles the low level hardware control by accessing control registers.

- M-HAL (Mediatek Hardware Abstraction Layer) - Provides the higher level API to application layer.

Besides M-HAL and HDL, MTK also provides the OS-HAL samples code which shows the M-HAL API invocation flow example. Please refer to the following for the software architecture diagram.

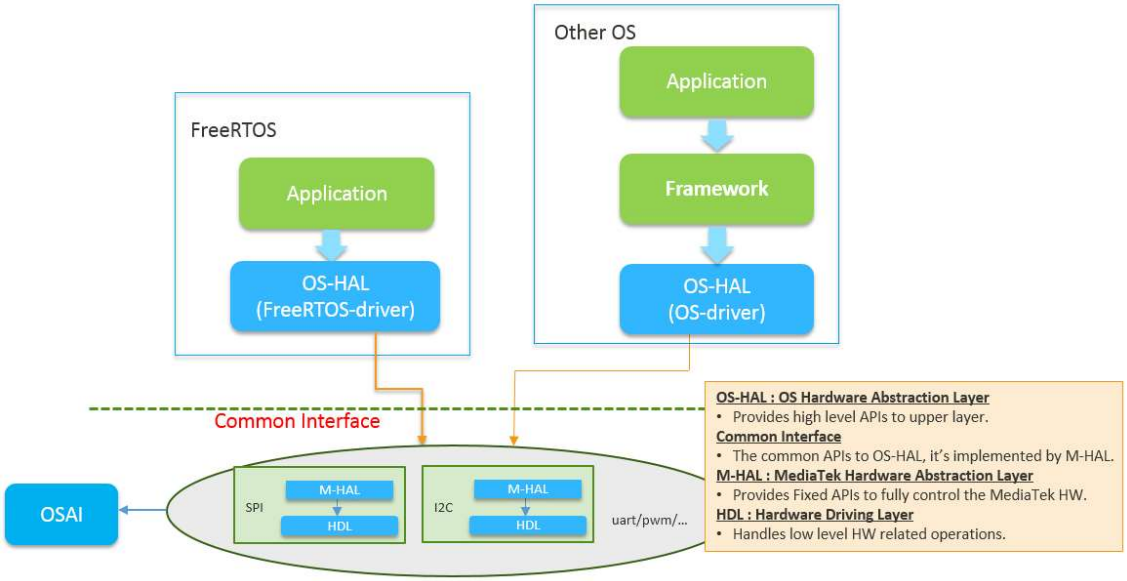

Figure 2. M-HAL Software Architecture

2.3.1 M-HAL API document

The completed M-HAL API document could be accessed the following URL:

- https://support.mediatek.com/AzureSphere/mt3620/M4_API_Reference_Manual/

2.3.2 M-HAL API Code

The M-HAL API source code is maintained on GitHub:

- https://github.com/MediaTek-Labs/mt3620_m4_software

2.3.3 Sample code

The sample code in CMake format is maintained on GitHub:

- https://github.com/MediaTek-Labs/mt3620_m4_software/tree/master/Sample_Code/BareMetal

2.4 Inter-Core Communication

MT3620 supports inter-core communication between high-level application on A7 and real-time application on M4. Please refer to the following for the detailed information:

- https://docs.microsoft.com/en-ca/azure-sphere/app-development/high-level-inter-app
- https://github.com/Azure/azure-sphere-samples/tree/master/Samples/IntercoreComms

2.5 MT3620 M4 Power Management

2.5.1 M4 Power Mode

The supported power modes are listed as the following table.

Table 1. Supported Power Modes

Subsys (CPU)	Power Mode	Note (Trigger/Control/Wakeup Source)
IO-CM4 MCUSYS	force-off	CA7-PLUTON mailbox/PLUTON-CM4 CR/CA7-PLUTON mailbox
	DeepSleep (DSLP)	Self-trigger(WFI)/mcusys sleep controller/WIC
	LegacySleep (LSLP)	Self-trigger(WFI)/mcusys sleep controller/WIC
	core clock gating	Self-trigger(WFI)/ARM IP/WIC

2.5.2 M4 Power Saving Control Scheme

CM4 subsystems are equipped with hardware low power control circuit (trigger by core IP side-band signal induced by WFI command).The M4 MCUSYS sleep controller will handshake with PLUTON CM4 in low power control flow. The sleep sequence is triggered by IO CM4 itself and controlled by HW automatically. The wakeup event relies on WIC natively. At SLEEP state, the keep alive clock is forced to 32 KHz LP clock.

2.5.2.1 How to enter sleep mode

The **Wait For Interrupt** instruction, **WFI**, causes M4 immediate entry to sleep mode unless the wake-up condition is true.
Please refer to the following URL of Cortex-M4 Devices Generic User Guide.
http://infocenter.arm.com/help/topic/com.arm.doc.dui0553b/DUI0553.pdf

2.5.2.2 How to wake up from sleep mode

The M4 wakes up only when it detects an exception with sufficient priority to cause exception entry.
Please refer to the following URL of Cortex-M4 Devices Generic User Guide.
http://infocenter.arm.com/help/topic/com.arm.doc.dui0553b/DUI0553.pdf

2.6 Memory Mapping

MT3620 address map consists of the region from 0x0 to 0x2FFF_FFFF that is private to each core's subsystem. Local SRAM and TCM shall live in this region for CM4 MCUs. The memory map table is shown below.

Table 2. IO CM4 Region Memory Mapping

Start address	End address	Size	Description	Comment
0x0000_0000	0x0000_0007	8B	1st jump instruction (8 bytes)	
0x0000_0008	0x000F_FFFF	1024K-8B	Reserved	Note *1

Start address	End address	Size	Description	Comment
0x0010_0000	0x0012_FFFF	192KB	TCM (Lower 32KB useable as XIP Cache)	Note *1
0x0013_0000	0x0FFF_FFFF	254MB	Reserved	Note *1
0x1000_0000	0x1FFF_FFFF	256MB	Alias address of flash XIP region	Note *2
0x2000_0000	0x200F_FFFF	1MB	CM4 CFG register	Note *1
0x2010_0000	0x20FF_FFFF	15MB	Reserved	
0x2100_0000	0x2100_FFFF	64KB	IO CM4 IRQ CFG register	
0x2101_0000	0x2101_FFFF	64KB	RGU low power control	
0x2102_0000	0x2102_FFFF	64KB	WDT register	
0x2103_0000	0x2103_FFFF	64KB	GPT timer register	
0x2104_0000	0x2104_FFFF	64KB	UART register	
0x2105_0000	0x2105_FFFF	64KB	Mailbox with CA7	
0x2106_0000	0x2106_FFFF	64KB	Mailbox with IO_CM4	
0x2107_0000	0x2107_FFFF	64KB	Infra Bus register	
0x2108_0000	0x2108_FFFF	64KB	GDMA	
0x2109_0000	0x210F_FFFF	512KB	Reserved	
0x2110_0000	0x21FF_FFFF	15MB	Reserved	
0x2200_0000	0x22FF_FFFF	16MB	IO CM4 subsystem SRAM	Note *3
0x2300_0000	0x2FFF_FFFF	208MB	Reserved	
0x3801_0000	0x3801_FFFF	64KB	GPIO/PWM_0 register	Note *4
0x3802_0000	0x3802_FFFF	64KB	GPIO/PWM_1 register	Note *4
0x3803_0000	0x3803_FFFF	64KB	GPIO/PWM_2 register	Note *4
0x3804_0000	0x3804_FFFF	64KB	GPIO/PWM_3 register	Note *4
0x3805_0000	0x3805_FFFF	64KB	GPIO/PWM_4 register	Note *4
0x3806_0000	0x3806_FFFF	64KB	GPIO/PWM_5 register	Note *4
0x3807_0000	0x3807_FFFF	64KB	ISU_0 register	Note *4
0x3808_0000	0x3808_FFFF	64KB	ISU_1 register	Note *4
0x3809_0000	0x3809_FFFF	64KB	ISU_2 register	Note *4
0x380A_0000	0x380A_FFFF	64KB	ISU_3 register	Note *4
0x380B_0000	0x380B_FFFF	64KB	ISU_4 register	Note *4
0x380C_0000	0x380C_FFFF	64KB	ISU_5 register	Note *4
0x380D_0000	0x380D_FFFF	64KB	I2S_0 register	Note *4
0x380E_0000	0x380E_FFFF	64KB	I2S_1 register	Note *4
0x380F_0000	0x380F_FFFF	64KB	I2S_0 data interface	Note *4
0x3810_0000	0x3810_FFFF	64KB	I2S_1 data interface	Note *4
0xE000_E000	0xE000_EFFF	4KB	NVIC, SYSTICK, FPU	Nested vectored interrupt controller System Control Space (SYSTICK) Floating-point unit (note 1)

Note *1: This region is inside MCU and will not appear at the local AHB bus.

Note *2: IO CM4 uses alias address for flash XIP region (not using global address) such that software of the two IO CM4 MCUs can use the same XIP alias address.

Note *3: Note that the IO CM4 subsystem SRAM can only be accessed by its own MCU.

Note *4: Note that the availability of IO/ ISU is assigned by Azure Sphere OS through manifest file definition

3. General

Please make sure to use the most recently issued product brief before initiating or completing a design. Customers are responsible for the design, operation, suitability and fitness for intended use of their own and customer's third-party customers' applications and products using MediaTek's products. By use of MediaTek's product, customer acknowledges and agrees to have in place appropriate and sufficient system and precaution measures and conduct all necessary testing and modification to minimize risks relating to customer's use of MediaTek's product within customer's product and application. Except as otherwise explicitly provided in the written agreement between MediaTek and customer, MediaTek makes no other warranties with respect to MediaTek's products to customer, expressed, implied, statutory or otherwise, and MediaTek disclaims any implied warranty, including without limitation any implied warranty of merchantability, non-infringement of third party intellectual property right, or fitness for a particular purpose.