CMOSTEK AN204

CMT2280F2/2281F2/2189B/2189C IDE Operation Guide

Overview

This document discusses the program development IDE operation guide for the CMT2280F2, CMT2281F2, CMT2189B and CMT2189C. As part of the CMOSTEK NextGenRF^{TM} product family, which covers short-range wireless communication chips including transmitters, receivers and transceivers, these 4 wireless MCUs use the same high-performance RISC core and support FLASH ROM for on-line debugging.

The product models covered in this document are shown in the table below.

Table 1. Product Models Covered in This Document

Product Model	Frequency Range	Modulation Method	Transmitting Power	Sensitivity	Operating Current	Configuration Method	Package
CMT2280F2	300 - 960 MHz	OOK	-	- 110 dBm	3.8 mA	MCU Configuring	SOP16
CMT2281F2	300 - 960 MHz	OOK	-	- 109 dBm	4.5 mA	MCU Configuring	SOP16
CMT2189B	240 - 960 MHz	OOK	+ 13 dBm	-	17.5 mA[1]	MCU Configuring	SOP14
CMT2189C	240 - 960 MHz	OOK/(G)FSK	+ 13 dBm	-	32.5 mA[2]	MCU Configuring	SOP14

Notes:

[1]. CMT2189B's transmitting power and transmitting current test conditions are: 433.92 MHz, CW mode (namely always in carrier transmitting mode), 50% Duty transmitting mode and transmitting current of 8.5 mA.

[2]. CMT2189C's transmitting power and transmitting current test conditions are: 433.92 MHz and FSK mode.

Table of Contents

1 Introduction to IDE

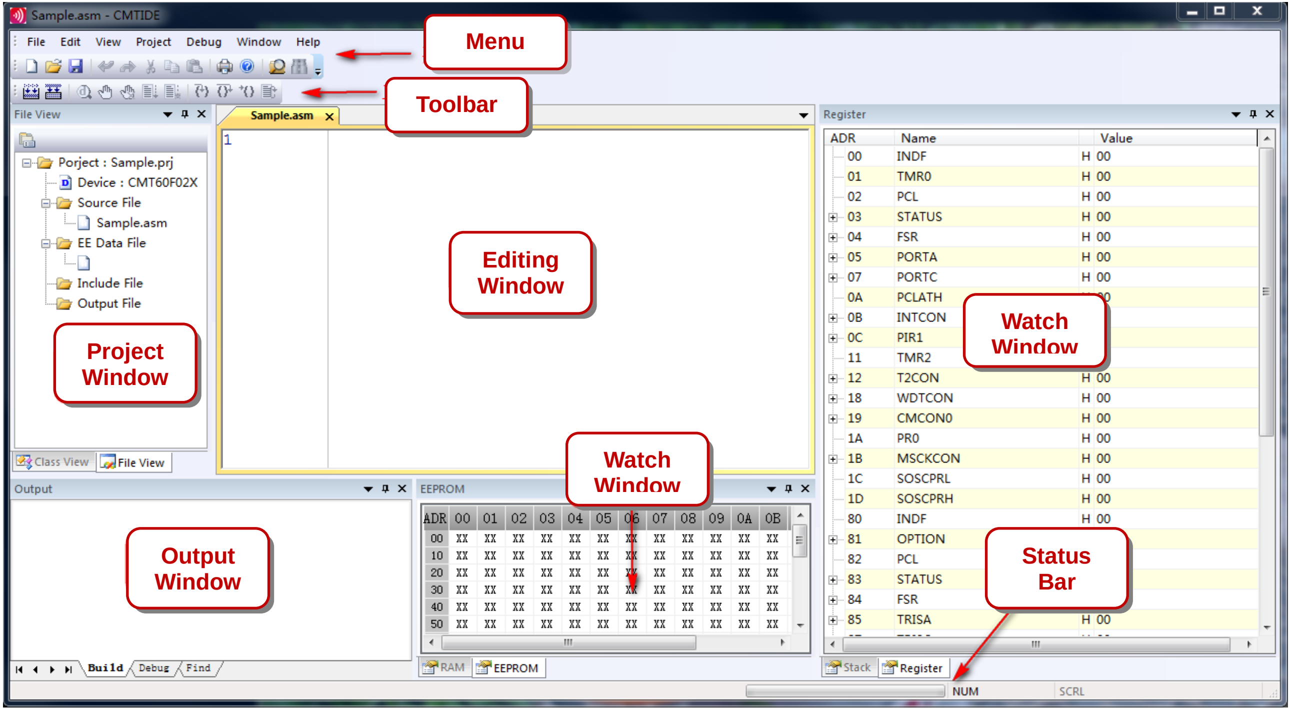
The IDE provides development software for the 4 wireless MCUs, CMT2280F2, CMT2281F2, CMT2189B, and CMT2189C. The major function is to debug and simulate the chip programs.

The IDE provides the following features.

- Create and edit source programs using the built-in editor.
- Compile the source code.
- Download and debug executable programs by connecting to the debugger.
- View variables in watch window.

2 Main Screen

As shown in the below figure, generally 4 basic windows are contained in the main screen including project window, editing window, output window and watch window, which are draggable, hovering, and docking windows except the editing window.

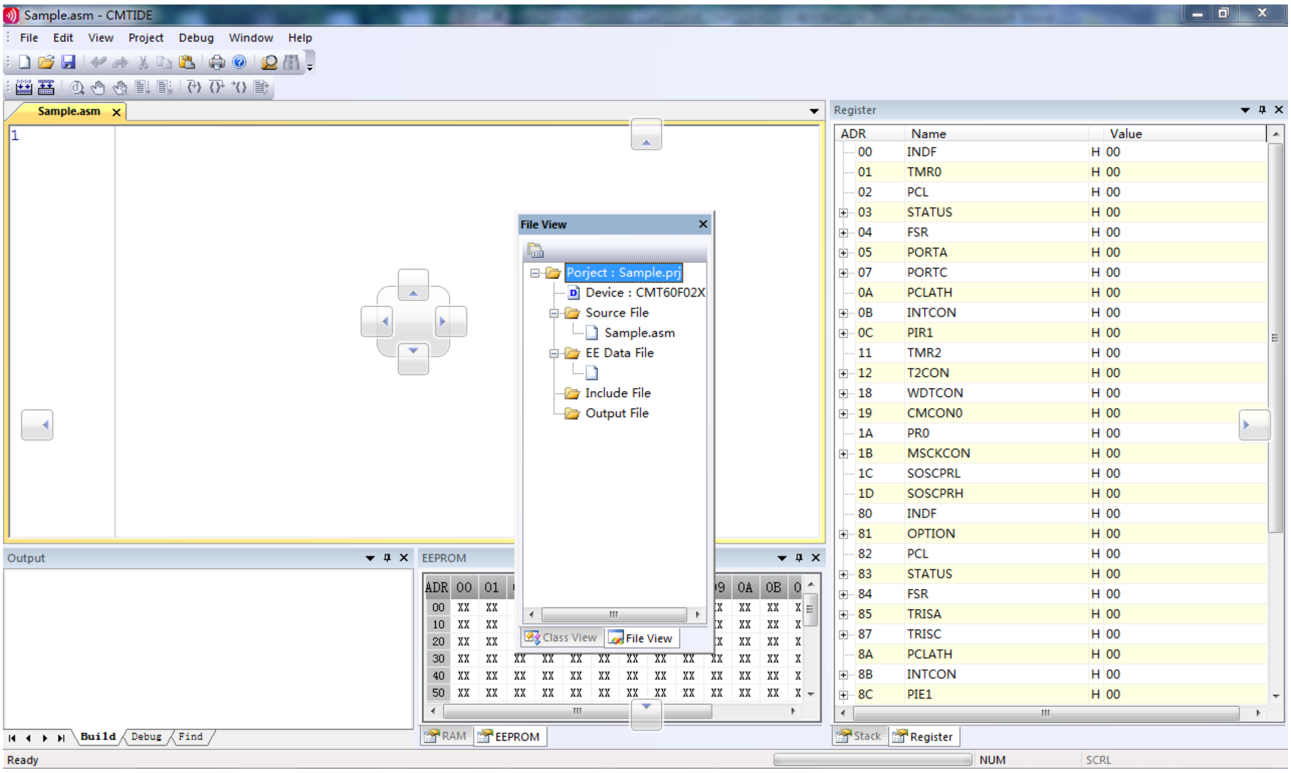


ADR	Name	Value
00	INDF	H 00
01	TMR0	H 00
02	PCL	H 00
03	STATUS	H 00
04	FSR	H 00
05	PORTA	H 00
07	PORTC	H 00
0A	PCLATH	
0B	INTCON	
0C	PIR1	
11	TMR2	
12	T2CON	H 00
18	WDTCON	H 00
19	CMCON0	H 00
1A	PR0	H 00
1B	MSCKCON	H 00
1C	SOSCPRL	H 00
1D	SOSCPRH	H 00
80	INDF	H 00
81	OPTION	
82	PCL	
83	STATUS	
84	FSR	
85	TRISA	H 00

ADR	00	01	02	03	04	05	06	07	08	09	0A	0B
00	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX
10	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX
20	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX
30	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX
40	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX
50	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX

Figure 1. Main Screen

For example, click and drag a window area, the docking arrow appears as shown in Figure 2. When a window is dragged into the direction specified by an arrow, it will dock in the position of the screen corresponding to the direction of the docking arrow.



ADR	Name	Value
00	INDF	H 00
01	TMR0	H 00
02	PCL	H 00
03	STATUS	H 00
04	FSR	H 00
05	PORTA	H 00
07	PORTC	H 00
0A	PCLATH	H 00
0B	INTCON	H 00
0C	PIR1	H 00
11	TMR2	H 00
12	T2CON	H 00
18	WDTCON	H 00
19	CMCON0	H 00
1A	PR0	H 00
1B	MSCKCON	H 00
1C	SOSCPRL	H 00
1D	SOSCPRH	H 00
80	INDF	H 00
81	OPTION	H 00
82	PCL	H 00
83	STATUS	H 00
84	FSR	H 00
85	TRISA	H 00
87	TRISC	H 00
8A	PCLATH	H 00
8B	INTCON	H 00
8C	PIE1	H 00

Figure 2. Dragging Window to Screen Position According to Docking Arrow Direction

2.1 Project Window

Project window contains 2 tabs, *File View* and *Class View*, as shown in the below figure. *File View* displays project information, including device name, source file, data file, reference file, and output file. Double click a file name under *Source File* or *Include File* to open it. *Class View* displays all labels after compiling completes. Double clicking a label in the editing window can locate the label in the source file.

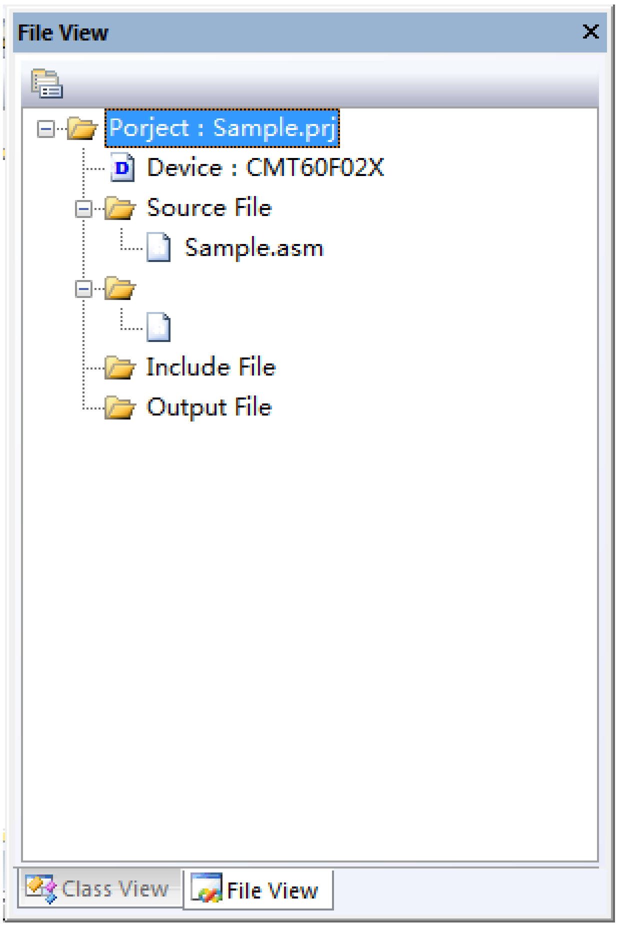


Figure 3. Tabs in Project Window

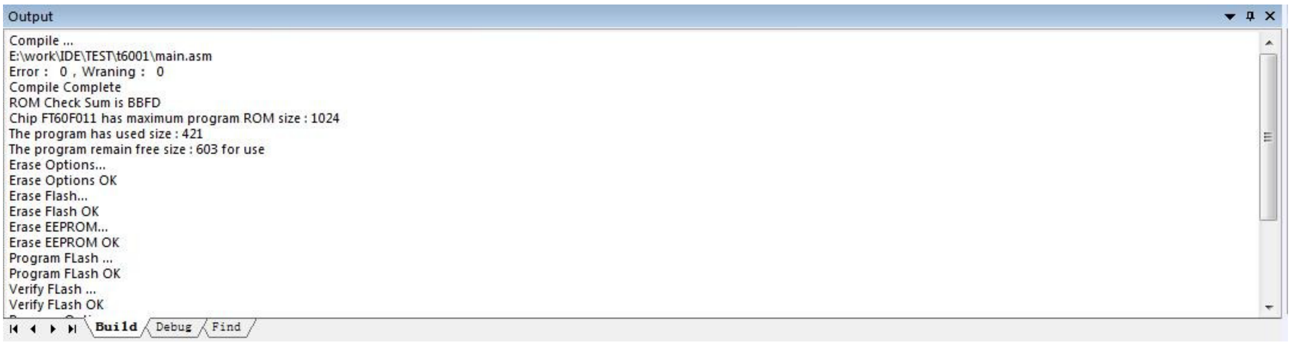
2.2 Editing Window

Users edit source files and reference files in editing window. The editor colors keywords, with commands in blue, immediate data in red and comments in green. Press *Ctrl* and scroll the middle mouse button to zoom in or out. It will exit debug mode automatically when file editing and saving is performed during debugging. Then users need to recompile and re-download before the next debugging. If a document open in the editing window is modified by an external editor, it will prompt to reload. When this occurs in debugging state, it will exit debugging as well. Users can debug it again only after recompiling and download.

2.3 Output Window

Output window contains 3 tabs, *Build, Debug* and *Find* as shown in the below figure (note that the tabs in the output window cannot be separated. while tabs in other windows can be separated or combined). *Build* tab displays the output information such as compiling and download. Compiling errors display the file containing the compiling error, line number and line statement of the error. Double-clicking can locate the line in the editing area.

To debug and watch a variable, users can input the variable to be watched in *Debug* tab, the variable and the corresponding address will be refreshed during the debugging. If the variable is a bit variable, the bit address and the corresponding bit value will be refreshed. *Find* tab lists the files and line numbers searched out. Double click a line can locate the line in editing window.



```
Compile ...
E:\work\IDE\TEST\t6001\main.asm
Error :  0 , Wraning :  0
Compile Complete
ROM Check Sum is BBFD
Chip FT60F011 has maximum program ROM size : 1024
The program has used size : 421
The program remain free size : 603 for use
Erase Options...
Erase Options OK
Erase Flash...
Erase Flash OK
Erase EEPROM...
Erase EEPROM OK
Program FLash ...
Program FLash OK
Verify FLash ...
Verify FLash OK
```

Figure 4. Output Window

2.4 Watch Window

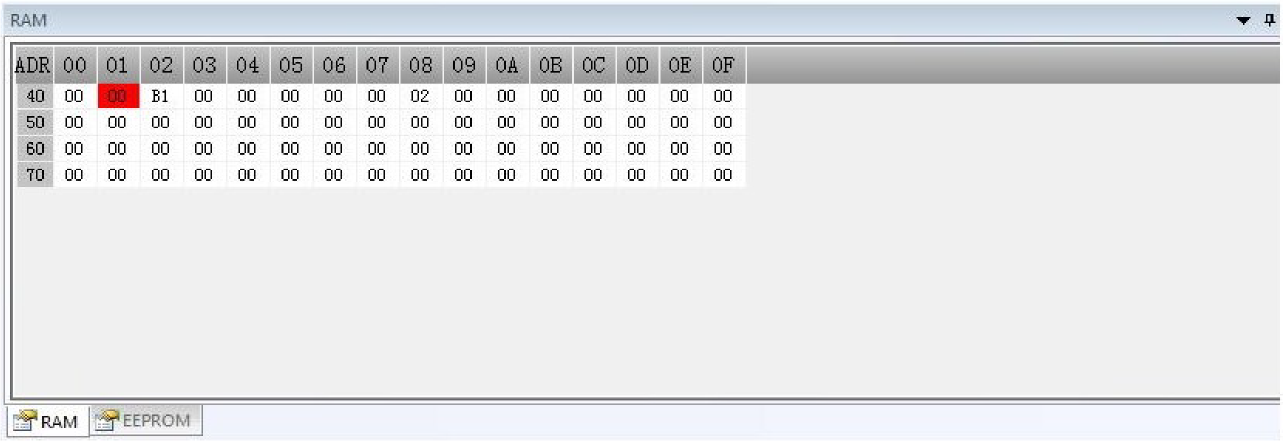
Watch window consists of 4 tabs, *RAM, EEPROM, STACK, and Register* as shown in the below figure.

RAM tab displays RAM value. During debugging, the RAM with change is marked in red. After starting the debugging, users can double click to modify a RAM value. After the modification, click anywhere in RAM tab, then the modified value will be written into the chip.

EEPROM tab displays the EEPROM value. In EEPROM tab, users can import data through the menu *Load EEPROM Data....* Users can also define EEPROM data in a source file through the DE command. Similar to RAM modification, It can also be edited in *EEPROM* tab. Double click and modify the EEPROM value, then click anywhere in the EEPROM tab ,the modified value will be written to the chip. Select the menu *Export EEPROM Data...* to export the EEPROM data. Note that if the program modifies the value of the EEPROM, the EEPROM data output after debugging may be different from the EEPROM data output when compiling completes.

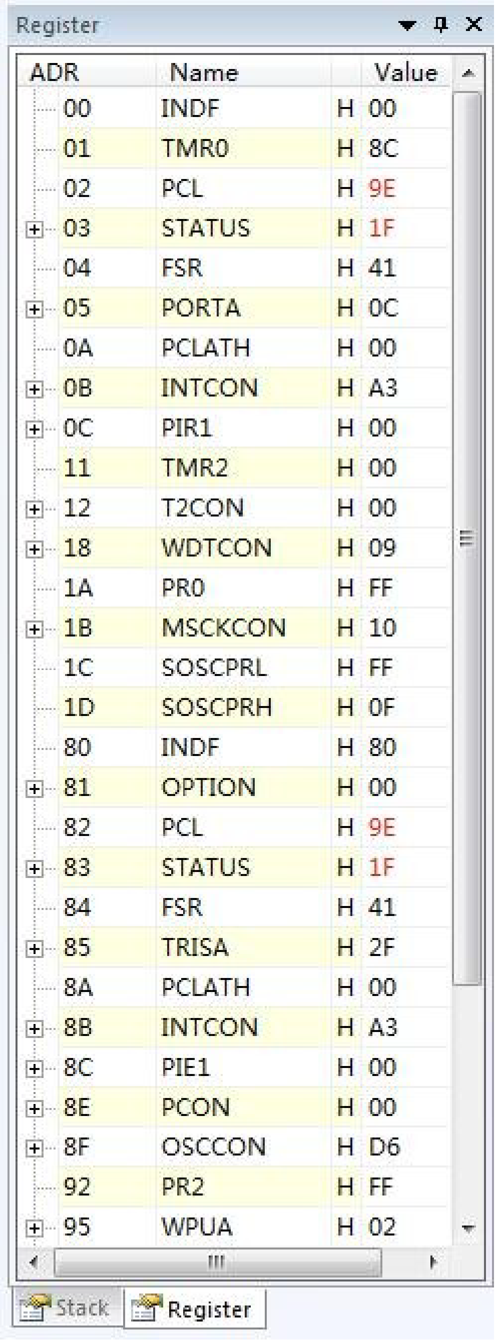
STACK tab shows stack values, which cannot be modified. It is only for information watch (watch stack pushing and popping information) purpose.

Register tab displays the SFR function specific register information, including the register address, register name and register value, and the data format of the displayed value (H for hexadecimal and B for binary). Click + to expand the display to show bit variable and the corresponding bit value. Click - to collapse bit variable display. The display is refreshed during debugging and registers in change are marked in red. During debugging, double click to modify the register value. Click anywhere in the *Register* tab, the modified value will be written to the chip.



ADR	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F
40	00	00	B1	00	00	00	00	00	02	00	00	00	00	00	00	00
50	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
60	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
70	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00

Figure 5. RAM and EEPROM Output Display



ADR	Name	Value
00	INDF	H 00
01	TMR0	H 8C
02	PCL	H 9E
03	STATUS	H 1F
04	FSR	H 41
05	PORTA	H 0C
0A	PCLATH	H 00
0B	INTCON	H A3
0C	PIR1	H 00
11	TMR2	H 00
12	T2CON	H 00
18	WDTCON	H 09
1A	PR0	H FF
1B	MSCKCON	H 10
1C	SOSCPRL	H FF
1D	SOSCPRH	H 0F
80	INDF	H 80
81	OPTION	H 00
82	PCL	H 9E
83	STATUS	H 1F
84	FSR	H 41
85	TRISA	H 2F
8A	PCLATH	H 00
8B	INTCON	H A3
8C	PIE1	H 00
8E	PCON	H 00
8F	OSCCON	H D6
92	PR2	H FF
95	WPUA	H 02

Figure 6. Stack and Register Watch

2.5 Menu, Toolbar and Status Bar

Menu contains all the operations. Toolbar corresponds to the commonly used menu items. Status bar shows the progress information, chip status and the debugger version number, as shown in the below figure.

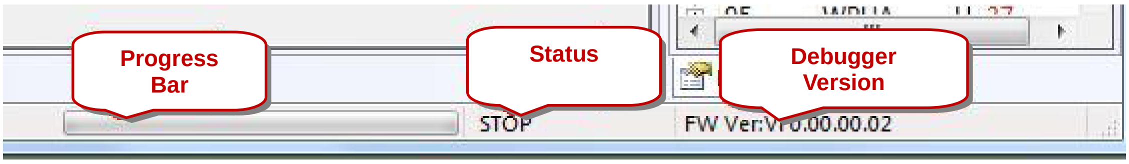
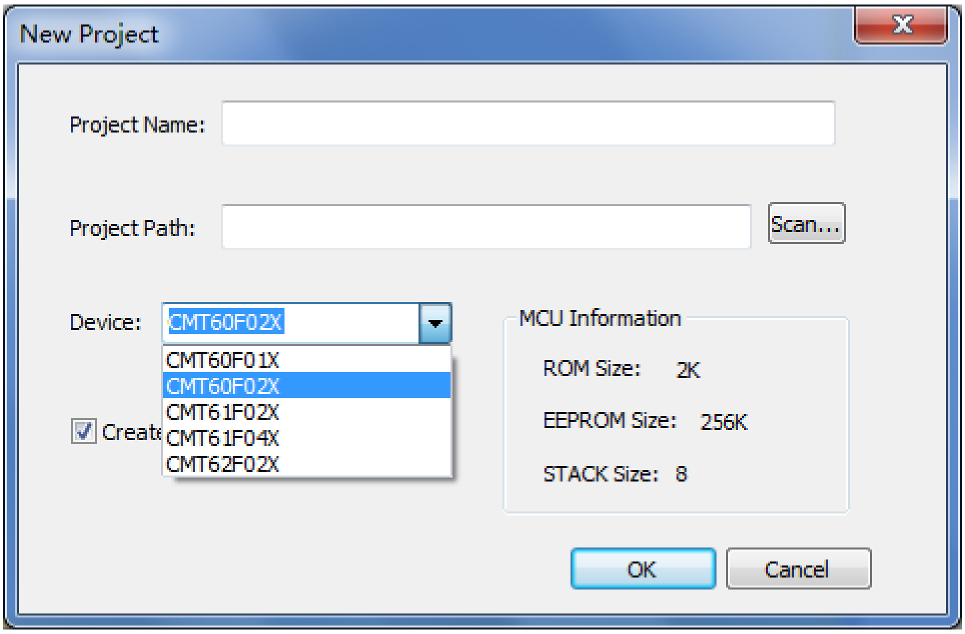


Figure 7. Status Bar

3 New Project

The IDE uses project management mode. Users need to build a project first by selecting *Project -> New Project* , then new project dialog is popped up as shown in the below figure. Fill in *Project Name*, select *Project Patch*, and *Device* (select CMT60F02X). If users need to create a project folder, check *Create directory for project*.


New Project

Project Name:

Project Path:

Device: CMT60F02X

☑ Create

MCU Information

ROM Size: 2K

EEPROM Size: 256K

STACK Size: 8

Figure 8. New Project

After new project operation completes, users need to add source files. 2 ways are available, 1) select *Project -> Add File...* 2) select *Source File* in project window, right click and select *Add File from the pop-up menu*....

Notes:

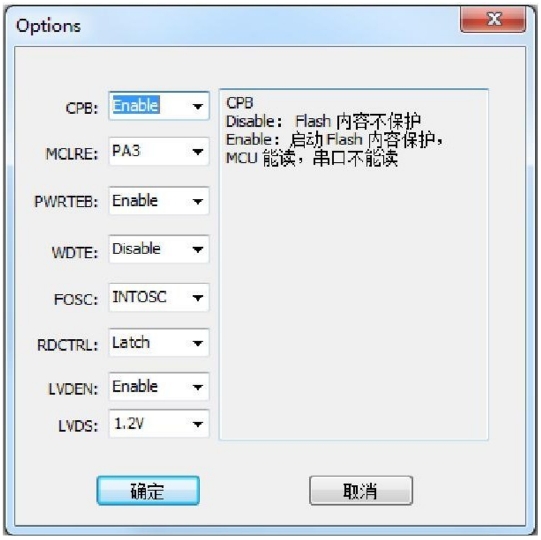
1. If the added file is not in the project directory, it will be copied to the project directory automatically.

   In addition to adding source files, users can add data files as well. 2 ways are available 1) select Project -> Load EEPROM Data... 2) in project window, select *EE Data File*, right click and select *Add File...* in the pop-up menu.

2. If the added file is not in the project directory, it will be copied to the project directory automatically. Include File and Output File are automatically generated after compilation and users do not need to add them.

4 Compiling

Select the menu *Debug -> Build (F7)* or click to compile. Before compiling, the *Option* dialog box will pop up as shown in the below figure. Select the corresponding option and click *OK* to start compiling. The compiling information will display in the *Build* window. In case of compiling error, double-click the error message to locate the error line. See Appendix 1 for detailed compiler descriptions.



Options

CPB: Enable

MCLRE: PA3

PWRTEB: Enable

WDTE: Disable

FOSC: INTOSC

RDCTRL: Latch

LVDEN: Enable

LVDS: 1.2V

CPB
Disable： Flash 内容不保护
Enable： 启动 Flash 内容保护，
MCU 能读，串口不能读

Figure 9. Compiling Options

5 EEPROM Settings

The EEPROM can be imported in 3 ways, 1) select menu *Project -> Load EEPROM Data...* 2) define data through DE command 3) double-click EEPROM tab to input data.

In case of menu import and DE data definition, the compiled DE definition will overwrite the data imported by a file. For method 3), users can only input after simulation starts by double-clicking the EEPROM. After users double-click the modification, it will be written to the EEPROM immediately with immediate reading back and display in the screen.

Select menu *Project -> Export EEPROM Data...* to export the data in the current EEPROM tab.

Note:

1. The EEPROM data is related to program operation.

6 Downloads

Select the menu *Debug -> Build All* (Ctrl+F7) to erase, download and reset. After successful download, the simulation toolbar becomes valid. When simulation starts and the file is modified and saved, the simulation toolbar becomes invalid. Users need recompile and download. If a program is not compiled before it is selected for download, it will perform compiling first, then erasing, downloading and resetting.

7 Simulation

7.1 Set Breakpoint

2 ways are available, 1) click to select the statement line to be set, select the menu *Debug -> BreakPoint (F9)* or click ✋. It will add a breakpoint if it is set for the first time. It will delete the breakpoint if it's set again. 2) click the line number in editing window to set a breakpoint, and click again to delete the breakpoint. Breakpoints are marked with a yellow circle. Currently, IDE-supported devices only allow one breakpoint to be set, so the last breakpoint is automatically deleted each time a new breakpoint is set.

7.2 Reset (Ctrl + F5)

Reset a device and the PC returns to 0. After reset completes, RAM, EEPROM, STACK, and registers are refreshed and chip status displays in status bar.

7.3 Step Into (F11)

Step Into is a single step run. For *Step Info*, when a *Call* instruction is executed, it enters a sub-function. When the execution completes, RAM, EEPROM, STACK, and registers are refreshed and chip status displays in status bar.

7.4 Step Over (F10)

Step Over is a single step run. For *Step Over* , when a *Call* instruction is executed, it does not enter a sub-function. When the execution completes, RAM, EEPROM, STACK, and registers are refreshed and chip status displays in status bar.

7.5 Run to Cursor (Ctrl + F10)

For *Run to Cursor*, users need to keep the cursor in the statement line where execution will stop, and the program will stop at the cursor position. If the cursor cannot be reached, the program will run at full speed. When execution completes, RAM, EEPROM, STACK, and registers are refreshed and chip status displays in status bar.

7.6 Go (F5)

Go will run to the breakpoint. Users need to set the breakpoint before execution, and the program runs to the breakpoint then stops. If the breakpoint cannot be reached, the program will run at full speed. When execution completes, RAM, EEPROM, STACK, and registers are refreshed and chip status displays in status bar.

7.7 Stop (Shift + F5)

It is to stop a program running at full speed. When the execution completes, RAM, EEPROM, STACK, and registers are refreshed and chip status displays in status bar.

8 Appendix 1 - Compiler Description

8.1 Variable Naming Rules

Variables must start with a letter, consisting of letters numbers and underscores, which are not case sensitive. Variable names cannot be keywords including instructions and pseudo instructions.

8.2 Digital Format

It supports binary, hexadecimal and decimal as follows:

Binary:

1) 0/1 characters ending with *B*, e.g. 00010110B.

2) 0/1 characters starting with *B*" and ending with ", e.g. B"00010110".

Hex:

1) Hexadecimal number ending with *H*, e.g. 1FH.

2) Hexadecimal number starting with *0x*, e.g. 0x1F.

3) Hexadecimal number starting with *H"* and ending with ", e.g. H"1F".

Decimal:

1) A number without suffix is a decimal, e.g.16.

2) Decimal numbers starting with ., e.g. .10.

8.3 Address Label

A label name follows the variable naming rules, and a same name cannot be defined repeatedly. A label can contain a suffix of colon *:* or does not contain colon *:.* All non-instructed and undefined single strings will be identified as labels. A label without a colon cannot be in the same line with an instruction. A label with a colon can be in the same line with an instruction.

8.4 Instruction

Table 2. Instructions

CMT	PIC	Instruction Cycle	Function	Operation	Status
BCRR, b	BCF F, b	1	Bit clear	0→R(b)	NONE
BSRR, b	BSF F, b	1	Bit set	1→R(b)	NONE
BTSC R, b	BTFSC F, b	1(2)	Bit test, skip if 0	Skip if R(b)=0	NONE
BTSS R, b	BTFSS F, b	1(2)	Bit test, skip if 1	Skip if R(b)=1	NONE
NOP	NOP	1	No operation	None	NONE
CLRWDT	CLRWDT	1	Clear WDT	0→WDT	/PF, /TF
SLEEP	SLEEP	1	Enter Sleep Mode	0→WDT，STOP OSC	/PF, /TF
STTMD	OPTION	1	Store W TO TMODE	W→TMODE	NONE
CTLIO R	TRIS R	1	Control IO direction reg	W→IODIRr	NONE
STR R	MOVWF F	1	Store W to reg	W→R	NONE
LDR R, d	MOVF F, d	1	Load reg to d	R→d	Z
SWAPR R, d	SWAPF F, d	1	Swap halves reg	[R(0-3)R(4-7)]→d	NONE
INCR R, d	INCF F, d	1	Increment reg	R+1→d	Z
INCRSZ R, d	INCFSZ F, d	1（2）	Increment reg, skip if 0	R+1→d	NONE
ADDWR R, d	ADDWF F, d	1	Add W and reg	W+R→d	C, HC, Z
SUBWR R, d	SUBWF F, d	1	Sub W from reg	R-W→d or R+/W+1→d	C, HC, Z
DECR R, d	DECF F, d	1	Decrement reg	R-1→d	Z
DECRSZ R, d	DECFSZ F, d	1（2）	Decrement reg, skip if 0	R-1→d	NONE
ANDWR R, d	ANDWF F, d	1	AND W and reg	R&W→d	Z
IORWR R, d	IORWF F, d	1	Inclu.OR W and reg	W|R→d	Z
XORWR R, d	XORWF F, d	1	Exclu.OR W and reg	W^R→d	Z
COMR R, d	COMF F, d	1	Complement reg	/R→d	Z
RRRR, d	RRF F, d	1	Rotate right reg	R(n)→R(n-1)，C→R(7)， R(0)→C	C
RLR R, d	RLF F, d	1	Rotate left reg	R(n)→R(n+1)，C→R(0), R(7)→C	C
CLRW	CLRW	1	Clear working reg	0→W	Z
CLRR R	CLRF F	1	Clear reg	0→R	Z
RETI	RETFIE	2	Return from interrupt	Stack→PC，1→GIE	NONE
RET	RETURN	2	Return from subroutine	Stack→PC	NONE
LCALL N	CALL k	2	Long CALL subroutine	N→PC, PC+1→Stack	NONE
LJUMP N	GOTO	2	Long JUMP address	N→PC	NONE
LDWI I	MOVLW k	1	Load immediate to W	I→W	NONE
ANDWI I	ANDLW k	1	AND W and imm	W&I→W	Z
IORWI I	IORLW k	1	Inclu.OR W and imm	W|I→W	Z
XORWI I	XORLW k	1	Exclu.OR W and imm	W^I→W	Z
RETW I	RETLW k	2	Return, place imm to W	Stack→PC，I→W	NONE
ADDWI I	ADDLW k	1	Add imm to W	W+I→W	C, HC, Z
SUBWI I	SUBLW k	1	Subtract W from imm	I-W→W	C, HC, Z

8.5 Pseudo Instruction

8.5.1 ORG

Format: ORG ADDR

Description: define the PC address. The ADDR cannot be smaller than the current PC or larger than the maximum PC.

Example:

```
ORG     0000H
Goto    START
ORG     0004H   ;interrupt entry
JUMP    INTtimer0
```

8.5.2 Include

Format: #Include<filename>, #include "*filename"*

Description: <*filename*> is a file in the system directory, *filename* is a file in the project directory, the file type can be a header file of *.H* or *.HIC*, or a source file of *.ASM*. A header file must be imported before the PC address is 0, and the source file can be imported anywhere in a file.

Example:

```
#INCLUDE <CMT2189B.inc>
#INCLUDE "LED.ASM"
```

8.5.3 EQU

Format: variable name EQU RAM address

Description the variable name follows the variable naming rules. The same variable name does not allow multiple addresses. Multiple variables are allowed to correspond to a same address. Note that the RAM address will not be checked if the variable is not used,.

Example:

```
LEDLEVEL EQU 0x40
```

8.5.4 DB

Format: variable name DB ?

Description: the variable name follows the variable naming rules. A same name cannot be defined repeatedly. The address is automatically assigned by the compiler without no need for specifying the corresponding RAM address. Note that the address is not assigned when a variable is not in use.

Example:

```
V1 DB?
```

8.5.5 DE

Format: DE Data0, Data1, ... Datan

Description: in the EEPROM data table, the address of the data table must be after 0x4100. The data after the DE table is arranged sequentially starting from the address defined by ORG. The number of data is unlimited, however it must be in the same line.

Example:

```
ORG         4110H
DE          0x10, 0x11, 0x12, 0x13, 0x14, 0x15, 0x16, 0x17
DE          0x18, 0x19, 0x1A, 0x1B, 0x1C, 0x1D, 0x1E, 0x1F
```

8.5.6 DBIT

Format: variable name DBIT ?

Description: it defines a bit variable that follows the variable naming rules. The address is automatically assigned by the compiler without specifying the corresponding RAM address. Note that the address is not assigned when a variable is not in use. The variables defined by this instruction can only be used in bit operation instructions including BCR, BSR, BTSC, BTSS, BCF, BSF, BTFSC and BTFSS.

Example:

```
V1_1 DBIT ?
```

8.5.7 Define

Format: #define identifier string

Description: macro definition without parameters. Macro definition is to use a macro name to represent a string, replace the macro name with the string, this is just a simple substitution, the string can contain any characters, can be constant, also Can be an expression, the preprocessor does not check it. If there is an error, it can only be found when compiling the source program that has been expanded by the macro.

Example:

```
#define Defname 1+5 ORG 0000H
......
LDWI Defname
```

8.5.8 MARCO

Format: macro name MARCO par1...part n

Macro content

ENDM

Description: macros with parameters. Macro names and parameter names follow the variable naming convention. A macro module with ENDM. No labels are allowed within the macro definition.

Example:

```
Delayms     macro a1, a2, a3
```

```
LDWI a1
STR DELAYCNT1
LDWI a2
STR DELAYCNT2
LDWI a3
STR DELAYCNT3
CALL DELAYLOOP
Endm

ORG 0000H
......
Delayms 0xF0, 0x49, 0x30
```

8.5.9 Ifdef

Format: ifdef conditional macro

```
; program 1
Else
;block 2 endif
```

Description: if a conditional macro is not equal to 0, block 1 is executed, otherwise block 2 is executed. Note that this instruction cannot be nested.

Example:

```
Ifdef       defname
    LJUMP RESTART_WDT DECRSZ DELAYCNT1, F
    LJUMP POWERDOWN_2SLOOP
Else
    DECRSZ DELAYCNT2, F
    LJUMP POWERDOWN_2SLOOP DECRSZ DELAYCNT3, F
    LJUMP POWERDOWN_2SLOOP
Endif
```

9 Appendix 2 - Chip Debugging Manual

9.1 Overview

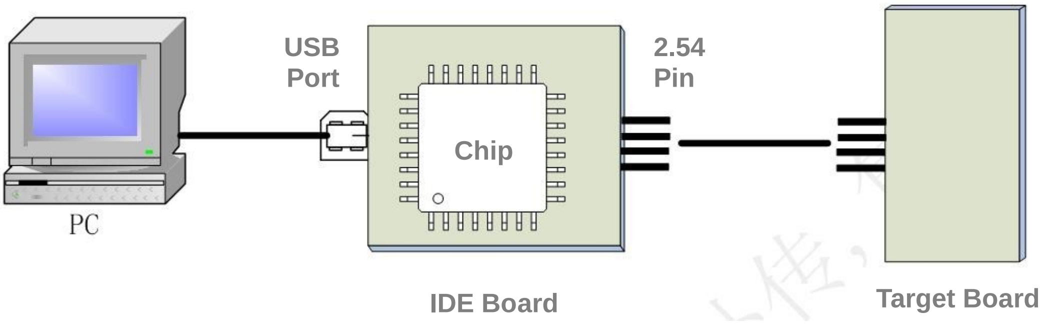


Figure 10. Development Environment Establishment Schematic Diagram

This appendix is to define and describe the interface between the IDE development board and the outside world in the product integration development environment, which covers related hardware and software interfaces.

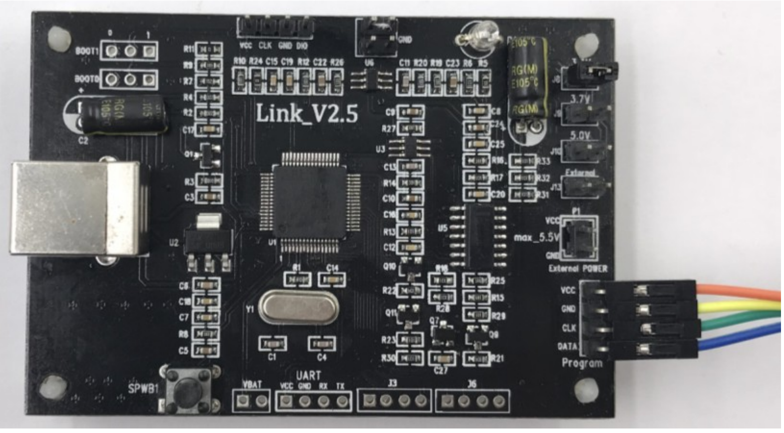


Figure 11. Debugging Board Diagram

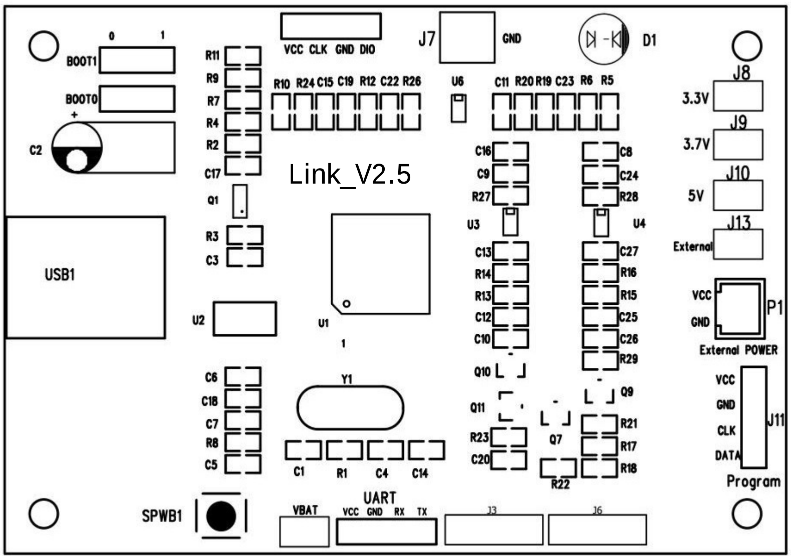


Figure 12. Debugging Board Top Marking Schematic Diagram

9.2 Interface Description

The interface description for Figure 12 is as follows.

- USB1: connection between USB port and PC.
- J11: program interface, which is the communication control port of a target chip with the connections from top to bottom listed in the below table.

Table 3. Connection Between Debugging Interface and Target Chip

J11 Interface	CMT2280F2	CMT2281F2	CMT2189B	CMT2189C
VCC	DVDD Pin13	DVDD Pin13	DVDD/AVDD Pin1/Pin6	VDD Pin1/Pin11
GND	GND Pin2	GND Pin3	GND Pin7	GND Pin2/Pin10/Pin13
CLK	PA0/ICSPCLK Pin9	PA0/ICSPCLK Pin9	PA0/ICSPCLK Pin12	PA0/ICSPCLK Pin8
DATA	PA1/ICSPDAT Pin10	PA1/ICSPDAT Pin10	PA1/ICSPDAT Pin13	PA1/ICSPDAT Pin9

P1: external power input

J7: download board GND

J8: target MCU 3.3 V power supply selection

J9: target MCU 3.7 V power supply selection

J10: target MCU 5 V power supply selection (only for CMT2281F2)

J13: external input power supply selection

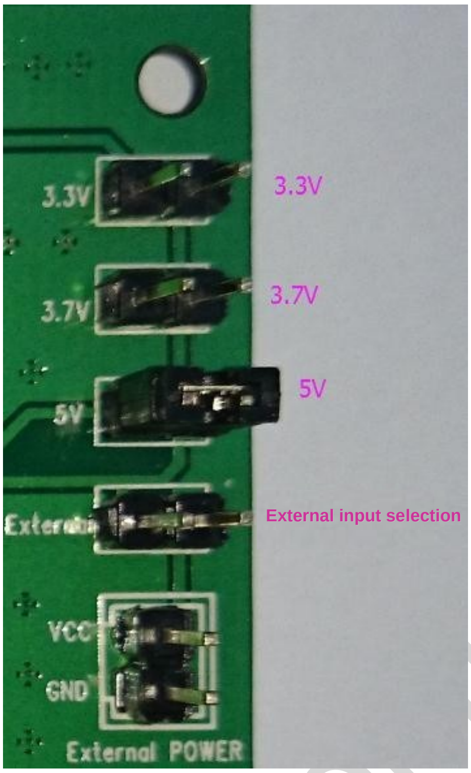


Figure 13. Debugging Board Power Supply Interface

Notes:

- For the 4 short-circuit points, J8, J9, J10 and J13, only one can be shorted.
- SPWB1: reset button for download board.
- D1: indicator light.

  Red LED (always on): power-on indication, at this time, the debugger does not communicate with PC successfully yet.

  Green LED (always on): the debugger and the PC is connected successfully.

  Yellow LED (always on): the debugger is in BOOT mode, waiting for the online firmware update.

  Yellow LED (flashing): the debugger is in operating.

9.3 Considerations

1. The connection between a PC host and the IDE development board is simple, which uses standard USB interface with no need for additional driver installation for the debugger. When a PC system is plugged in for the first time, the driver will be installed automatically and then users can use the debugger normally.
2. The debugger board is powered by the PC via USB 5 V port. When a target chip/board is debugged using the debugger board, it can be powered directly by the debugger. The power supply voltage can be switched and adjusted through a short-circuit cap. It supports the voltages of 3.3 V, 3.7 V and 5 V and only one of them can be shorted at the same time. If users need external power supply, it requires to disconnect the target board power then connect the external power supply to the external power interface of the debugger board. Then the power is provided to the target board through the VCC of Program interface. The maximum input voltage of the external power interface cannot exceed 12 V.

3. Each time when switching power or replacing target chip, users need to press the reset button on the debugger board to reconnect the target chip. When the LED on the debugger board turns green, it means communication with the host computer is successful. At this time, users can operate the host computer for debugging.

4. Upon each power up, the host PC will detect the firmware version number. If the firmware version number is found too low, the system will jump into boot mode. At this time, the yellow LED light is always on and the host will update firmware automatically. Please don't disconnect the USB during this processing. In case it is disconnected or firmware update fails, it requires users to press the reset button or re-plug USB to let it enter boot mode again for firmware re-update.

10 Revise History

Table 4. Revise History Records

Version No.	Chapter	Description	Date
1.0	All	Initial version	2017-11-25
1.1	2	Update some IDE screenshots	201-11-28

11 Contacts

CMOSTEK Microelectronics Co., Ltd. Shenzhen Branch
Address: 2/F Building 3, Pingshan Private Enterprise S.T. Park, Xili, Nanshan District, Shenzhen, Guangdong, China

Tel:	+86-755-83231427
Post Code:	518057
Sales:	sales@cmostek.com
Supports:	support@cmostek.com
Website:	www.cmostek.com

The information furnished by CMOSTEK is believed to be accurate and reliable. However, no responsibility is assumed for inaccuracies and specifications within this document are subject to change without notice. The material contained herein is the exclusive property of CMOSTEK and shall not be distributed, reproduced, or disclosed in whole or in part without prior written permission of CMOSTEK. CMOSTEK products are not authorized for use as critical components in life support devices or systems without express written approval of CMOSTEK. The CMOSTEK logo is a registered trademark of CMOSTEK Microelectronics Co., Ltd. All other names are the property of their respective owners.