DM6814/DM6914 Driver for Windows 98/NT4/2000/XP

version 4.2

User's Manual

RTD Embedded Technologies, Inc.
"Accessing the Analog World"®

SWM-640010007
Rev. B

ISO9001 and AS9100 Certified

RTD Embedded Technologies, INC.
103 Innovation Blvd.
State College, PA 16803-0906

Phone: +1-814-234-8087
FAX: +1-814-234-5218

E-mail
sales@rtd.com
techsupport@rtd.com

web site
http://www.rtd.com

Revision History

07/29/2003	revision A released Documented for ISO9000
09/10/2003	revision B released Version 4.2 released Company name changed Windows 95 support removed

DM6814/DM6914 Driver for Windows 98/NT4/2000/XP
Published by:

RTD Embedded Technologies, Inc.
103 Innovation Blvd.
State College, PA 16803-0906

Copyright 2003 RTD Embedded Technologies, Inc.
All rights reserved
Printed in U.S.A.

The RTD Logo is a registered trademark of RTD Embedded Technologies. cpuModule and utilityModule are trademarks of RTD Embedded Technologies. PS/2, PC/XT, PC/AT and IBM are trademarks of International Business Machines Inc. MS-DOS, Windows, Windows 98, Windows NT, Windows 2000 and Windows XP are trademarks of Microsoft Corp. PC/104 is a registered trademark of PC/104 Consortium. All other trademarks appearing in this document are the property of their respective owners.

Table of Contents

TABLE OF CONTENTS **4**

INTRODUCTION **5**

INSTALLING OF THE DRIVER AND EXAMPLE PROGRAMS **6**

THE DM6814 BOARD DRIVER **7**

INSTALLING A NEXT DM6814 BOARD IN THE SYSTEM **7**

USING WINDOWS REGISTRY KEYS **8**

THE DRIVER API FUNCTIONS **10**

USING THE DRIVER WITHOUT THE WINDOWS-GUI **10**

DRIVER API FUNCTION GROUPS **10**

DRIVER INITIALIZATION FUNCTIONS 10
GENERAL BOARD CONTROL FUNCTIONS 10
DIGITAL I/O FUNCTIONS 10
INTERRUPT HANDLING 10
USER TIMER-COUNTER FUNCTIONS 11
INCREMENTAL ENCODER FUNCTIONS 11
ERROR HANDLING FUNCTIONS 11

ALPHABETICAL DRIVER API FUNCTIONS REFERENCE **12**

EXAMPLE PROGRAMS REFERENCE **29**

WIN32 WINDOWS APPLICATIONS 29
WREAD GUI (MFC) EXAMPLE 30
WINTRPTS GUI (MFC) EXAMPLE 30
WTIMERS GUI (MFC) EXAMPLE 30
LOOPBACK GUI (MFC) EXAMPLE 31

LIMITED WARRANTY **32**

Introduction

The DM6814/DM6914 Windows 98/NT/2000/XP driver was designed for programmers who write Windows-based application programs with using the RTD’s DM6814 or DM6914 board. DM6814 and DM6914 are the same boards with different interface connector. Later we will refer both boards as DM6814.

The driver provides an Application Programming Interface with a lot of function calls to perform all the data acquisition tasks of the board users.

The board driver based on the BlueWater System's WinRT device driver kit.

There are example programs to demonstrate the various board features and the usage of the driver API. The example programs are written in Microsoft Visual C++ ver. 6.0.

Installing of the Driver and Example Programs

Before installing the driver and example program files, you need to install the DM6814 board in your PC. Please follow the instructions of the manufacturer, how to install the board in a computer.

To install the drivers for the DM6814 under Windows 98/2000/XP you need the installation diskette 0. The readme.txt file on this diskette describes the necessary installation steps.

For Windows NT4.0 the installation diskette 0 is not needed.

On the installation diskette 1 you can found the **Setup.exe** program, which installs on your PC the DM6814 board driver and example programs. The setup program automatically detects your operating system and installs the appropriate files on your PC.
After starting the setup, please follow the instructions on the screen to install the programs. You can select the directory where to install the files.
The setup also adds to the 'Start menu' under the 'Programs' folder of your Windows system the 'RTD Embedded Technologies' folder. It contains shortcuts to the example programs, the driver configuration utility and the readme.txt file.

The example programs will be installed in the target directory under the 'Examples' folder. This folder contains the example program sources for the DM6814 board, and the project files for Microsoft Visual C++ users to rebuild the programs. The example programs are compiled with the Microsoft Visual C++ ver. 6.0. In case of different version of Visual C is installed on your PC, please rebuild the executable files.

Uninstallation: you can uninstall the DM6814 driver and example programs under the 'Control panel' with the 'Add/Remove Programs' tool.

The DM6814 board driver

The RTD's DM6814 board driver handles the hardware through the BlueWater's WinRT driver. The WinRT driver provides the low-level access to the board, and the RTD's Drvr6814.dll provides the device API for the programmers, and communicates with the hardware through the WinRT driver.

The RTD board drivers has multi-board feature. It means that it is possible to use more than one board in the system.

The base address and the interrupt channel of the DM6814 board can be set by DIP switch. If you are using multiple boards in your system, you need assign different addresses to each board. When you install a WinRT driver for the board, you must select the Base Address and IRQ, and during the operating of the board must be used this settings.

Each board requires a unique WinRT device to handle the hardware. The maximum number of devices on Windows NT is 32, on Windows 98 is 10.

During the installation the Setup.exe program set the default driver values:
Board 1 : 768 (0x300) base address, IRQ 5

If these settings are not proper to your PC configuration, you can modify the settings with the **Drvr_cfg.exe** program (Windows NT4).
In case of Windows 98/2000/XP the operating system uses a WDM driver and it can be configured in the Control Panel/System/[Hardware]/Device Manager.

Installing a next DM6814 board in the system

It is possible to use more than one board in the system using the RTD's data acquisition board drivers. Please follow the next steps to install a multi-board – system:

- Install the first board with all software and make sure that everything is working well.
- Place the second board in an empty slot in the PC, and switch on the computer.
  Windows NT4.0: After the system startup start the installation program and answer 'YES' to the next board installation question. Restarting the system with the two-board example program is possible to test the working of both boards. Please note that during the second board installation the driver and example program files will be refreshed thus you need to save the important changes before starting the install.
  Windows 98/2000/XP: You need only the installation diskette 0 to install the next boards. Follow the steps described in the readme.txt file on diskette 0.

Using Windows registry keys

The RTD board drivers are using the next registry keys:

Windows NT 4.0:

```
HKEY_LOCAL_MACHINE
|
| -- System
     |
     | -- CurrentControlSet
          |
          | -- Services
               |
               | -- WinRT
                    |
                    | -- WinRTdev0       (one per device)
                    |     |
                    |     | -- Parameters
                    |           |
                    |           | -- Section0

                    | -- WinRTdev1
                    |     |
                    |     | -- Parameters
                    |           |
                    |           | -- Section0
                    |
```

Windows 98 Enum key path:

```
HKEY_LOCAL_MACHINE
|
| -- Enum
```

Windows 98 Class key path:

```
HKEY_LOCAL_MACHINE
|
| -- System
     |
     | -- CurrentControlSet
          |
          | -- Services
               |
               | -- Class
                    |
                    | -- WinRTDevices
```

Windows 2000/XP device class information:

```
HKEY_LOCAL_MACHINE
|
| -- System
     |
     | -- CurrentControlSet
          |
          | -- Control
               |
               | -- Class
                    |
                    | -- {D695ED6A-630D-4D83-D8B-F1F0AC107AD0}
                    |
                    | -- 0001
```

Windows 2000/XP Plug and Play devices are enumerated under the Enum key:

```
HKEY_LOCAL_MACHINE
|
| -- System
   |
   | -- CurrentControlSet
        |
        | -- Enum
             |
             | -- PCI
```

The Driver Api Functions

The resources on the DM6814 board can be accessed from Windows through the driver API (Application Programming Interface) functions. The executable code of these functions is located in the Drvr6814.dll file. To write applications using the API functions you must include the Reg6814.H header file, and link the program with the Drvr6814.lib import library file.
In the example programs you can find different examples how to use the driver.

Using the Driver without the Windows-GUI

This driver is based on the Win32 system, and can run only under Windows. However, the user, who is not familiar with the Windows Graphical User Interface, can use the driver API functions in Win32 console applications, which has an MS-DOS like text-mode interface.

In console applications is not necessary to use the Windows graphical environment, but all the driver's API functions are accessible.
The Microsoft Visual C++ compiler supports writing console applications.

Driver API Function Groups

Driver Initialization Functions

OpenBoard6814
CloseBoard6814

General Board Control Functions

InitBoard6814
ClearBoard6814

Digital I/O Functions

ReadDIO6814
WriteDIO6814
SetDIODirection6814

Interrupt Handling

ClearIRQ6814
ClearBoardIRQ6814
EnableIrq6814
EnableBoardIRQ6814
EnableIrqSharing6814
InstallCallbackIRQHandler6814
InstallCounterIRQHandler6814

RemoveIRQHandler6814
IsEPLD1IRQ6814
IsEPLD2IRQ6814
IsEPLD3IRQ6814
IsP14IRQ6814
LoadIRQRegister6814
ReadBoardIRQStatus6814
SetIRQPolarity6814

User Timer-Counter Functions

ClockDivisor6814
ClockMode6814
DoneTimer6814
EnableClearInput6814
ReadTimerCounter6814
SetUserClock6814

Incremental Encoder Functions

ClearChip6814
DisableIncEnc6814
EnableIncEnc6814
HoldIncEnc6814
LoadIncEnc6814
ReadIncEnc6814
SelectRegister6814

Error Handling Functions

GetErrorStatus6814

Alphabetical Driver API Functions Reference

C

ClearBoardIRQ6814

void ClearBoardIRQ6814(RTDHANDLE hBoard);

Description:

This routine is used to clear the board IRQ circuitry and status bit.
Parameters:

hBoard:	device handle

ClearChip6814

void ClearChip6814(RTDHANDLE hBoard, uchar8 SelectedChip);

Description:

This routine clears the selected Incremental Encoder chip.

Parameters:

hBoard:	device handle
SelectedChip:	0, 1, 2 .....

ClearIRQ6814

void ClearIrq6814(RTDHANDLE hBoard, uchar8 SelectedChip);

Description:

This routine clears the selected DIO chips IRQ status bit.

Parameters:

hBoard:	device handle
SelectedChip:	0, 1, 2 .....

ClockDivisor6814

void ClockDivisor6814(RTDHANDLE hBoard, uchar8 Timer, uint16 Divisor);

Description:
This routine is used to set the divisor of a designated counter on the 8254 programmable interval timer (PIT). This procedure assumes that the counter has already been set to receive the least significant uchar8 (LSB) of the divisor followed by the most significant uchar8 (MSB).

Parameters:

hBoard:	device handle
Timer:	0,1,2
Divisor:	0 - 65535

ClockMode6814

void ClockMode6814(RTDHANDLE hBoard, uchar8 Timer, uchar8 Mode);

Description:

This routine is used to set the mode of a designated counter in the 8254 programmable interval timer (PIT).

Parameters:

hBoard:	device handle
Timer:	0,1,2
Mode:	0,1,2,3,4,5

CloseBoard6814

BOOL CloseBoard6814 (LONG DEVICE_NO, LPSTR szBuf);

Description:

This routine is used to close board driver.

Parameters:

DeviceNumber:	WinRT device number to load
szBuf:	message buffer

Return value:

TRUE on success, FALSE on error

DisableIncEnc6814

void DisableIncEnc6814(RTDHANDLE hBoard, uchar8 SelectedChip);

Description:

This routine disables the selected DIO chips Incremental Encoder counter.

Parameters:

hBoard:	device handle
SelectedChip:	0, 1, 2 .....

DoneTimer6814

void DoneTimer6814(RTDHANDLE hBoard);

Description:

Initialize the timers for high speed to ensure the immediate load.

Parameters:

hBoard:	device handle

EnableBoardIRQ6814

void EnableBoardIRQ6814(RTDHANDLE hBoard, uchar8 Enable);

Description:

This routine enables the board IRQ for the DM6814.

Parameters:

hBoard:	device handle
Enable:	0 = disabled 1 = enabled

EnableClearInput6814

void EnableClearInput6814(RTDHANDLE hBoard, uchar8 SelectedChip, uchar8 Enable);

Description:

This routine enables P0.2 as an input line used to clear the up/down counter.

Parameters:

hBoard:	device handle
SelectedChip:	0, 1, 2 .....
Enable:	0 = disabled 1 = enabled

EnableIncEnc6814

void EnableIncEnc6814(RTDHANDLE hBoard, uchar8 SelectedChip);

Description:

This routine enables the selected DIO chips Incremental Encoder counter.

Parameters:

hBoard:	device handle
SelectedChip:	0, 1, 2 .....

EnableIrq6814

void EnableIrq6814(RTDHANDLE hBoard, uchar8 SelectedChip, uchar8 Enable);

Description:

This routine enables the selected DIO chips interrupt.

Parameters:

hBoard:	device handle
SelectedChip:	0, 1, 2 .....
Enable:	0 = Disabled
	1 = Enabled

EnableIrqSharing6814

void EnableIRQSharing6814(RTDHANDLE hBoard, uchar8 Enable);

Description:

This routine enables and disables the interrupt sharing feature of the board.

Parameters:

hBoard:	device handle
Enable:	1 = Disabled
	0 = Enabled

GetErrorStatus6814

BOOL GetErrorStatus6814 (RTDHANDLE hBoard, LONG *ErrorCode ,
LPSTR ErrorString);

Description:

This routine is used to get the error code and string in case of errors.
The function clears the last error code.

Parameters:

hBoard:	device handle
ErrorCode:	Error Code
ErrorString:	Return error string

Return Value:

TRUE if error occured
FALSE otherwise

HoldIncEnc6814

void HoldIncEnc6814(RTDHANDLE hBoard, uchar8 SelectedChip);

Description:

This routine latches the selected DIO chips Incremental Encoder counter.

Parameters:

hBoard:	device handle
SelectedChip:	0, 1, 2 .....

InitBoard6814

void InitBoard6814(RTDHANDLE hBoard);

Description:

This routine should always be called first.
This clears the board and variables the driver uses.

Parameters:

hBoard:	device handle

InstallCallbackIRQHandler6814

BOOL InstallCallbackIRQHandler6814 (RTDHANDLE hBoard, isr_t handler);

Description:

This routine is used to install IRQ handler function.

Parameters:

hBoard:	device handle
isrT handler:	user callback function

Return value:

TRUE on success, FALSE on error

InstallCounterIRQHandler6814

BOOL InstallCounterIRQHandler6814(RTDHANDLE hBoard, IRQSetup *irqsetup);

Description

This routine is used to install IRQ handler function.

Parameters:

hBoard:	device handle
irqsetup:	return parameters to caller

Return value:

TRUE on success, FALSE on error

IsEPLD1IRQ6814

uchar8 IsEPLD1IRQ6814(RTDHANDLE hBoard);

Description:

This routine checks the EPLD1 IRQ status bit.

Parameters:

hBoard: device handle

Returns:

1 if IRQ has been generated
0 if no IRQ

IsEPLD2IRQ6814

uchar8 IsEPLD2IRQ6814(RTDHANDLE hBoard);

Description:

This routine checks the EPLD2 IRQ status bit.

Parameters:

hBoard: device handle

Returns:

1 if IRQ has been generated
0 if no IRQ

IsEPLD3IRQ6814

uchar8 IsEPLD3IRQ6814(RTDHANDLE hBoard);

Description:

This routine checks the EPLD3 IRQ status bit.

Parameters:

hBoard: device handle

Returns:

1 if IRQ has been generated
0 if no IRQ

IsP14IRQ6814

uchar8 IsP14IRQ6814(RTDHANDLE hBoard);

Description:

This routine checks the P14 IRQ status bit.

Parameters:

hBoard: device handle

Returns:

1 if IRQ has been generated
0 if no IRQ

LoadIRQRegister6814

void LoadIRQRegister6814(RTDHANDLE hBoard, uchar8 Value);

Description:

This routine loads the interrupt register with one write operation.

Parameters:

hBoard:	device handle
Value:	0 - 7

LoadIncEnc6814

void LoadIncEnc6814(RTDHANDLE hBoard, uchar8 SelectedChip, uint16 Start);

Description:

This routine loads the starting value in the selected DIO chips Incremental Encoder counter.

Parameters:

hBoard:	device handle
SelectedChip:	0, 1, 2 .....
Start:	0 – 65535

OpenBoard6814

BOOL OpenBoard6814(int num, int device_id, LPSTR szBuf, BoardConfig *boardconfig);
Description:

This routine is used to open board driver.

Parameters:

num: the number of the board
device_id: (equals 0x6814)
szBuf: Message buffer
BoardConfig: a pointer variable to return the board configuration (interrupt No and the device handle (number of appropriate WinRT device)

Return value:

TRUE on success, FALSE on error

ReadBoardIRQStatus6814

uchar8 ReadBoardIRQStatus6814(RTDHANDLE hBoard);

Description:

This routine returns the IRQ status from the board.

Parameters:

hBoard: device handle

Returns:

8 bit unsigned integer

ReadDIO6814

uchar8 ReadDIO6814(RTDHANDLE hBoard, uchar8 SelectedChip);

Description:

This routine reads the selected DIO chips port.

Parameters:

hBoard: device handle
SelectedChip: 0, 1, 2 .....

Returns:

8 bit unsigned integer

ReadIncEnc6814

uint16 ReadIncEnc6814(RTDHANDLE hBoard, uchar8 SelectedChip);

Description:

This routine reads the selected DIO chips Incremental Encoder counter.

Parameters:

hBoard: device handle
SelectedChip: 0, 1, 2 .....

Returns:

16 bit unsigned integer

ReadTimerCounter6814

uint16 ReadTimerCounter6814(RTDHANDLE hBoard, uchar8 Timer);

Description:

This routine is used to read the contents of the selected timer/counter.

Parameters:

hBoard:	device handle
Timer:	0, 1, 2

Returns:
uint16

RemoveIRQHandler6814

void RemoveIRQHandler6814(RTDHANDLE hBoard);

Description:

This routine is used to uninstall IRQ handler function.

Parameters:

hBoard:	device handle

SetDIODirection6814

void SetDIODirection6814(RTDHANDLE hBoard, uchar8 SelectedChip, uchar8 Dir0,
uchar8 Dir1);

Description:

This routine sets the selected DIO chips bit I/O direction.

Parameters:

hBoard:	device handle
SelectedChip:	0, 1, 2 .....
Dir0: (P0.0)	0 = In
	1 = Out
Dir1: (P0.1)	0 = In
	1 = Out

SelectRegister6814

void SelectRegister6814(RTDHANDLE hBoard, uchar8 SelectedChip, uchar8 Select);

Description:

This routine picks the different registers on the selected Incremental Encoder chip.

Parameters:

hBoard:	device handle
SelectedChip:	0, 1, 2 .....
Select:	0 = Clear mode
	1 = Hold mode
	2 = Digital I/O
	3 = reserved

SetIRQPolarity6814

void SetIRQPolarity6814(RTDHANDLE hBoard, uchar8 Polarity);

Description:

This routine sets the P14 interrupt to use the positive edge or negative edge.

Parameters:

hBoard:	device handle
Polarity:	0 = positive edge
	1 = negative edge

SetUserClock6814

double64 SetUserClock6814(RTDHANDLE hBoard, float32 Rate);

Description:

This routine sets the user timer counter. The way this routine is written, the maximum clock rate is 10 KHz. This can easily be changed by adjusting the divisor for clock 0.

Parameters:

hBoard:	device handle
Rate:	10 KHz - .306 Hz

Returns:

The actual clock frequency that is programmed.

WriteDIO6814

void WriteDIO6814(RTDHANDLE hBoard, uchar8 SelectedChip, uchar8 Data);

Description:

This routine writes the selected DIO chips port.

Parameters:

hBoard:	device handle
SelectedChip:	0, 1, 2 .....
Data:	0 - 3

Example Programs Reference

Win32 Windows applications

Name	Feature	Remarks
wread	Sample program that demonstrates how to read the incremental encoder inputs.	The incremental encoder counters are sampled and displayed periodically.
wintrpts	Sample program that demonstrates how to use the timer as an interrupt source.	The incremental encoder counters are sampled and displayed on interrupt.
wtimers	Sample program that demonstrates how to program the user timer.	The counters are programmed to count the elapsed time in seconds.

WREAD *GUI (MFC) Example*

This simple program samples from the incremental encoder inputs in a software loop.

The digital I/O port bit 0 and bit 1 are used to simulate the output of an incremental encoder. So this program demonstrates the using of the digital I/O, too.

Connect the digital output bit 0 to INC ENC CHA, bit 1 to INC ENC CHB to count the pulses. The counter value of the incremental encoder is displayed on the screen numerically.

WINTRPTS *GUI (MFC) Example*

This program sets up the User Timer to generate interrupts at a selected rate.

An interrupt service routine is provided that simply reads the incremental encoders each time the interrupt is activated. The value is displayed on the screen as a foreground process. Although the ISR in this program is trivial, all the ingredients for powerful interrupt driven data acquisition are included.

The digital I/O port bit 0 and bit 1 are used to simulate the output of an incremental encoder.

Connect the digital output bit 0 to INC ENC CHA, bit 1 to INC ENC CHB to count the pulses.

WTIMERS *GUI (MFC) Example*

Sample program demonstrating how to program the 8254 programmable interval timers (PITs).

Counter 0 is programmed to run at 200 Hz. Counter 1 is used to count these pulses.

The result is converted into seconds and printed on the screen.

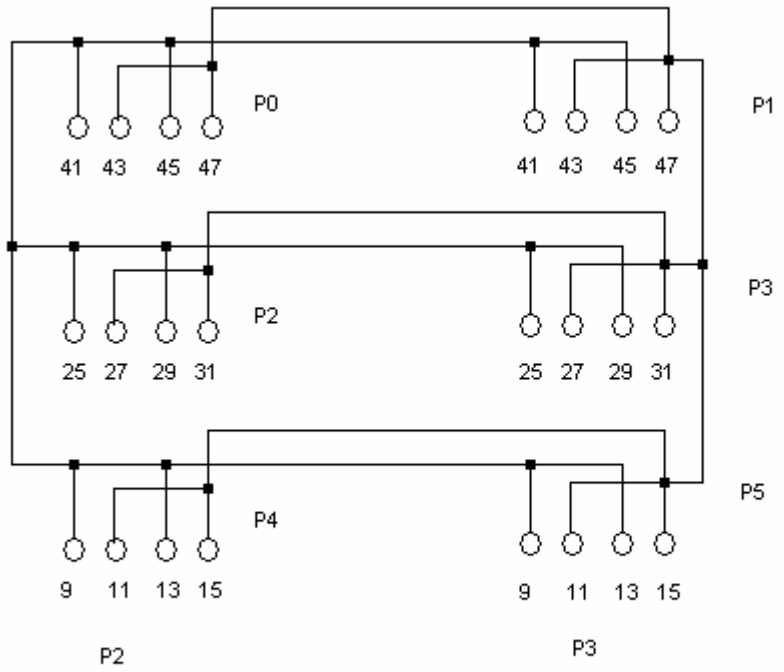
This sample program demonstrates how to use the Digital I/O. The program sets up the DIO to allow the even numbered DIO ports to communicate with the ports on other chips.

For proper results you should connect the following pins:

P0.3:41,P0.1:45,P1.3:41,P1.1:45,P2.3:25,P2.1:29,P3.3:25,P3.1:29,P4.3:9,P4.1:13,P5.3:9, P5.1:13

and

P0.2:43,P0.0:47,P1.2:43,P1.0:47,P2.2:27,P2.0:31,P3.2:27,P3.0:31,P4.2:11,P4.0:15,P5.2:11, P5.0:15



This wiring configuration allows loop-back testing.

The result is printed on the screen.

Limited Warranty

RTD Embedded Technologies, Inc. warrants the hardware and software products it manufactures and produces to be free from defects in materials and workmanship for one year following the date of shipment from RTD Embedded Technologies, INC. This warranty is limited to the original purchaser of product and is not transferable.

During the one year warranty period, RTD Embedded Technologies will repair or replace, at its option, any defective products or parts at no additional charge, provided that the product is returned, shipping prepaid, to RTD Embedded Technologies. All replaced parts and products become the property of RTD Embedded Technologies. Before returning any product for repair, customers are required to contact the factory for an RMA number.

THIS LIMITED WARRANTY DOES NOT EXTEND TO ANY PRODUCTS WHICH HAVE BEEN DAMAGED AS A RESULT OF ACCIDENT, MISUSE, ABUSE (such as: use of incorrect input voltages, improper or insufficient ventilation, failure to follow the operating instructions that are provided by RTD Embedded Technologies, "acts of God" or other contingencies beyond the control of RTD Embedded Technologies), OR AS A RESULT OF SERVICE OR MODIFICATION BY ANYONE OTHER THAN RTD Embedded Technologies. EXCEPT AS EXPRESSLY SET FORTH ABOVE, NO OTHER WARRANTIES ARE EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, ANY IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE, AND RTD Embedded Technologies EXPRESSLY DISCLAIMS ALL WARRANTIES NOT STATED HEREIN. ALL IMPLIED WARRANTIES, INCLUDING IMPLIED WARRANTIES FOR MECHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE, ARE LIMITED TO THE DURATION OF THIS WARRANTY. IN THE EVENT THE PRODUCT IS NOT FREE FROM DEFECTS AS WARRANTED ABOVE, THE PURCHASER'S SOLE REMEDY SHALL BE REPAIR OR REPLACEMENT AS PROVIDED ABOVE. UNDER NO CIRCUMSTANCES WILL RTD Embedded Technologies BE LIABLE TO THE PURCHASER OR ANY USER FOR ANY DAMAGES, INCLUDING ANY INCIDENTAL OR CONSEQUENTIAL DAMAGES, EXPENSES, LOST PROFITS, LOST SAVINGS, OR OTHER DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE PRODUCT.

SOME STATES DO NOT ALLOW THE EXCLUSION OR LIMITATION OF INCIDENTAL OR CONSEQUENTIAL DAMAGES FOR CONSUMER PRODUCTS, AND SOME STATES DO NOT ALLOW LIMITATIONS ON HOW LONG AN IMPLIED WARRANTY LASTS, SO THE ABOVE LIMITATIONS OR EXCLUSIONS MAY NOT APPLY TO YOU.

THIS WARRANTY GIVES YOU SPECIFIC LEGAL RIGHTS, AND YOU MAY ALSO HAVE OTHER RIGHTS WHICH VARY FROM STATE TO STATE.

RTD Embedded Technologies, Inc.
103 Innovation Blvd.
State College PA 16803-0906
USA
Our website: www.rtd.com