
TN8000.16

Technical Note

Coding with RIDE Quick Start

Author : G. Guye
For further information, please contact

XEMICS SA
E mail: info@xemics.com
Web: www.xemics.com

CONTENTS

TABLE OF CONTENTS

1	Introduction	3
1.1	Purpose of the document.....	3
1.2	Requirements.....	3
1.3	About the given example	3
2	Setting-up the project.....	4
3	Ride working directory structure	5
4	Basic files for the XE88LCXX	5
4.1	Template files.....	5
4.2	Files included with ride	5
5	Starting your application coding.....	6

TABLE OF FIGURES

Figure 1	Renaming the project.....	4
Figure 2	Renaming the application file.....	4
Figure 3	Changing the target type	4
Figure 4	Working directory structure.....	5
Figure 5	Code example.....	6

TABLE OF TABLES

Table 1	Types definition.....	5
---------	-----------------------	---

1 INTRODUCTION

1.1 PURPOSE OF THE DOCUMENT

This document describes how to start programming a XE8000 family microcontroller using RIDE. It gives an example of the structure to set in your project.

1.2 REQUIREMENTS

To complete the points described in this document you will need :

- A PC with RIDE installed.
- A zip file named template.zip
- This document.

1.3 ABOUT THE GIVEN EXAMPLE

Example files: The files can be found on the website under www.xemics.com/support

The zip file available with this technical note is an example of the way to set up your project. It provides templates of the basic files that are in nearly all projects.

It is not mandatory to follow this philosophy exactly, this is only an example for the users that want to start to use the XE88LCXX quickly.

This code is made to ease the start of a project with RIDE, and is not memory space saving oriented but more « user friendly » oriented.

To save space you can reduce the « InitUART() » function to it's strict minimum. This function is a switch made to take into account all the possible configurations of the UART, once you have decided on one, you can suppress all the other cases.

2 SETTING-UP THE PROJECT

1. Unzip the file to your working directory
2. Rename the directory from « template » to your project name.
3. Rename the template.prj file to your project name

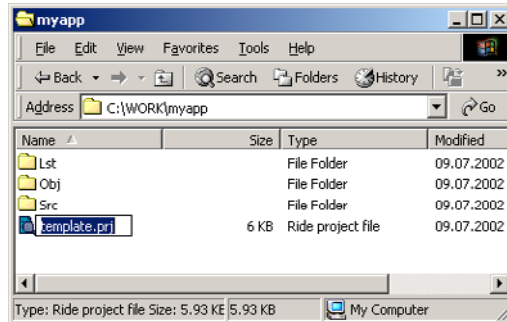


Figure 1 Renaming the project

4. Then you can launch RIDE and open the created project.
5. In RIDE you may change the application file to yours (right click on the name).

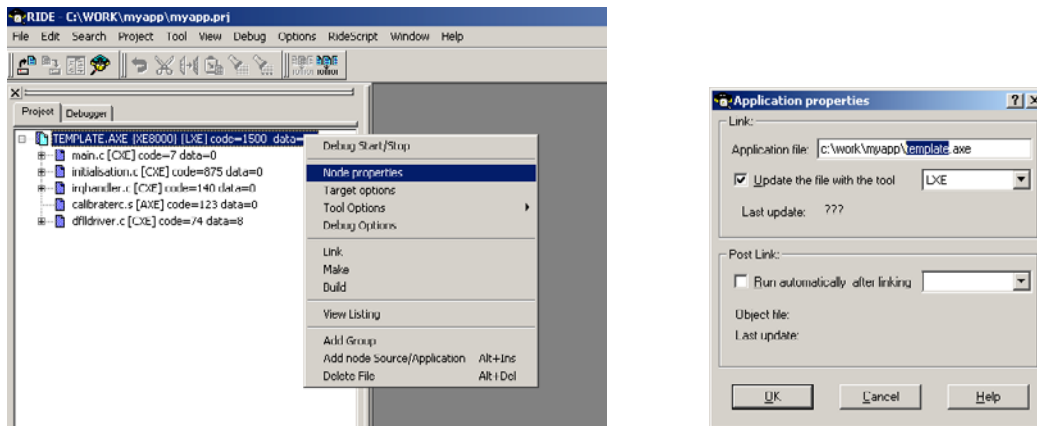


Figure 2 Renaming the application file

6. When completed you need to set the target

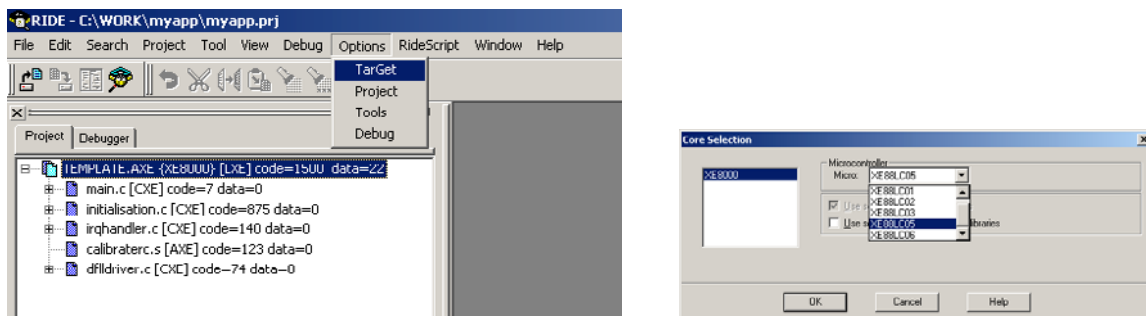


Figure 3 Changing the target type

7. Now you are ready to start coding!

3 RIDE WORKING DIRECTORY STRUCTURE

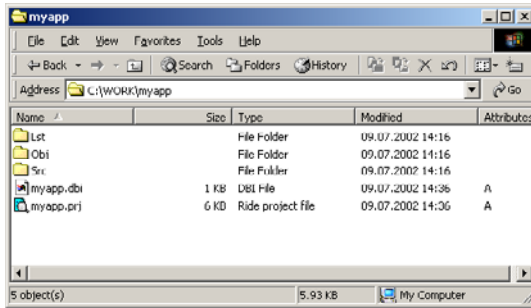


Figure 4 Working directory structure

There are three directories in your working directory

- Lst : All the *.lst files generated by the tool are placed here.
- Obj : All the *.lst files generated by the tool are placed here.
- Src : All your source files *.c; *.h *.s must be placed here.

4 BASIC FILES FOR THE XE88LCXX

4.1 TEMPLATE FILES

- Main.c : This file contains the entry point of your application, here you can manage the initialization of the microcontronroller, and call all the functions of your application.
- Initialisation.c & .h : This file contains all the initialization functions that may be called by the main function (manually), the .h file contains the prototypes of these functions.
- IRQHandler.c : This file contains predefined empty functions for every interrupt generated by the XE88LCXX
- DFLLDriver.c & .h : This file contains the DFLL function, function that allows the user to calibrate the RC oscillator using the XTAL with a precision of 2% (see TN8000.09 on our website for more information). You don't need to modify the file it is called by the UART initialization.
- CalibrateRC.s : This file contains assembler code for calibrate the RC oscillator (see DFLLDriver.c&h). You don't need to modify the file it's called by the DFLLDriver.c file.
- Globals.h : In this file you may include all the other header files that you want to be visible in the project. In this file you may declare global variables and definitions for code generation.

4.2 FILES INCLUDED WITH RIDE

- Types.h : This file contains definition of types, the types defined are the following :

Standard type name	Defined type name	Comment
bool (unsigned char)	false/true – 0/1	8 bit quantity
unsigned char	_U8	8 bit quantity
char	_S8	8 bit signed quantity
unsigned short	_U16	16 bit quantity
short	_S16	16 bit signed quantity
unsigned long	_U32	32 bit quantity
long	_S32	32 bit signed quantity
float	_F24	24 bit float quantity
double	_F32	32 bit float quantity

Table 1 Types definition

- xe88lcxx.h : This file contains all the registers definitions of the chosen target.
Note : This file is loaded by the user in the globals.h file under « Soft generation control target type ».

5 STARTING YOUR APPLICATION CODING

Now you can start coding your application. Following is an example of coding for the main application :

```

/*****
** File      : main.c
**
** Version   : V 1.0
**
** Written by : G.Guye
**
** Date      : 09-07-2002
**
** Project   : Myapp
**
** Changes   :
**
** Description : Main program
**
#include "Globals.h"
/*****
** Global variables declaration
**
_U16 variable1 = 0xFFFF;
_F32 variable2 = 3.0;

/*****
** main : Main program function
**
** In   : -
** Out  : -
**
/*****
int main (void){

    InitPortA();           // Initialize PORTA
    InitPortB();           // Initialize PORTB
    InitPortC();           // Initialize PORTC
    InitCounters();       // Initialize Counter

// Initialisation of the UART peripheral

/*****
** Parameters :
** Baudrate : 115200, 57600, 38400, 19200, 9600, 4800, 2400,
**           1200,600, 300          bauds
** Data Len : 1 = 8, 0 = 7          bits
** Parity    : 0 = ODD, 1 = EVEN
** ParityEn  : Enables or disables the parity
** rcFactor  : 1, 2, 4, 8, 16 times the default rc frequency
**
/*****
InitUART(19200, 1, 0, 1, 4);

// Infinite loop that call all the other functions
while(1){
    myfunction1();
    myfunction2();
}
return 0;
}

```

Figure 5 Code example

In the example above, the main function calls first all the initialization functions, then goes in an infinite loop that calls all the other functions.

Note : All the peripherals basic configuration can be changed in the initialization.c file.

©XEMICS 2002

All rights reserved. Reproduction in whole or in part is prohibited without the prior written consent of the copyright owner. The information presented in this document does not form part of any quotation or contract, is believed to be accurate and reliable and may be changed without notice. No liability will be accepted by the publisher for any consequence of its use. Publication thereof does not convey nor imply any license under patent or other industrial or intellectual property rights.