Application Note

Document No.: AN1132

G32R501 Zidian Application Note

Version: V1.0

1 Introduction

This application note introduces relevant content of G32R5xx Zidian, including basic introduction of Zidian and software design examples of the applications.

Contents

2 Zidian Overview

Zidian is a mathematical extension instruction set designed for the G32R5xx series real-time control MCU, and it can effectively improve the performance of mathematical operations.

2.1 ICAU and FCAU

In G32R5xx, there are two types of instruction extensions: CDE and FPCDE.

For extension of CDE instructions, the instructions operate on general-purpose registers, and some specific instructions are implemented in Zidian to improve the performance of integer operations, including:

- Calculation requiring SIMD: FFT, complex operations, etc.
- CRC algorithm

This set of instructions is called integer computation acceleration unit (ICAU).

For extension of FPCDE instructions, the instructions only operate on floating-point registers, and some specific instructions are implemented in Zidian to improve the performance of floating-point operations, including:

- Trigonometric function
- Square root
- Division

This set of instructions is called floating-point computation acceleration unit (FCAU).

2.2 CDE Built-in Functions

By introducing the standardized built-in functions of CDE as part of the ARM C language extension, Tables 1 and 2 list the built-in functions supported by Zidian.

Table 1 Built-in Functions Supported by ICAU

Instruction type	Built-in functions
CX2	uint32_t __arm_cx2(int coproc, uint32_t n, uint32_t imm);
CX2A	uint32_t __arm_cx2a(int coproc, uint64_t acc, uint32_t n, uint32_t imm);
CX2DA	uint64_t __arm_cx2da(int coproc, uint64_t acc, uint32_t n, uint32_t imm);
CX3	uint32_t __arm_cx3(int coproc, uint32_t n, uint32_t m, uint32_t imm);
CX3D	uint64_t __arm_cx3d(int coproc, uint32_t n, uint32_t m, uint32_t imm);

Instruction type	Built-in functions
CX3DA	uint64_t __arm_cx3da(int coproc, uint64_t acc, uint32_t n, uint32_t m, uint32_t imm);
CX2	uint32_t __arm_cx2(int coproc, uint32_t n, uint32_t imm);

Table 2 Built-in Functions Supported by FACU

Instruction type	Built-in functions
VCX2	uint32_t __arm_vcx2(int coproc, uint32_t n, uint32_t imm);
VCX3	uint32_t __arm_vcx3(int coproc, uint32_t n, uint32_t m, uint32_t imm);

Through these built-in functions, G32R5xx renames these instructions in Zidian. Table 3 takes the VCX3 instruction as an example and lists the renamed VCX3 functions in Zidian.

Table 3 VCX3 Rename Function

Instruction	Zidian function
VCX3 0, Sd, Sn, Sm, #0x0	DIVF32 Sd, Sn, Sm
VCX3 0, Sd, Sn, Sm, #0x1	QUADF32 Sd, Sn, Sm
VCX3 0, Sd, Sn, Sm, #0x2	DIVF32_ATAN2 Sd, Sn

3 Using Zidian to Accelerate Computation

3.1 zidian_math.h

In the file “zidian_cde.h”, the built-in functions are renamed through macro definition. If it is necessary to use Zidian for regular mathematical function operation in the driverlib library, the file “zidian_math.h” shall be included. Please refer to 3.3.1 for details.

The file “zidian_math.h” repackages the commonly used mathematical functions. Table 4 lists the mathematical functions packaged in the “zidian_math.h” file.

Table 4 zidian_math Function

Zidian function	Description
__sinpuf32	Calculate the sine value
__sin	Normalize it to the [0, 2π) interval, and then calculate the sine value
__cospuf32	Calculate the cosine value
__cos	Normalize it to the [0, 2π) interval, and then calculate the cosine value
__atanpuf32	Calculate the arc tangent value
__atan	Calculate the arc tangent value
__mpy2pif32	Single-precision floating-point multiplication
__div2pif32	Single-precision floating-point division
__sqrtf32	Square root of single-precision floating-point number
__divf32	Floating-point division
__quadf32	Calculate the quadrant values of X and Y
__divf32_atan2	Calculate the ratio of X to Y
__atan2puf32	atan2
__atan2	atan2

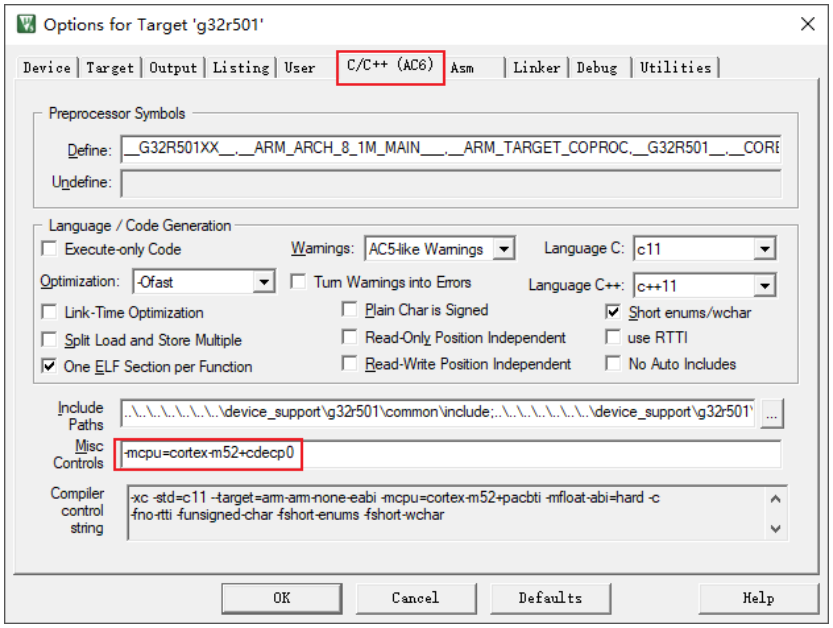
3.2 Compiler Support

At present, the SDK provides two environments of MDK-ARM and IAR, and the support of Zidan is slightly different under different compilers.

3.2.1 MDK-ARM (AC6)

In the option configuration card, add “-mcpu=cortex-m52+cdecp0” to the Misc Controls tab under the C/C++ (AC6) tab to enable the support.

Figure 1 AC6 Enabling CDE

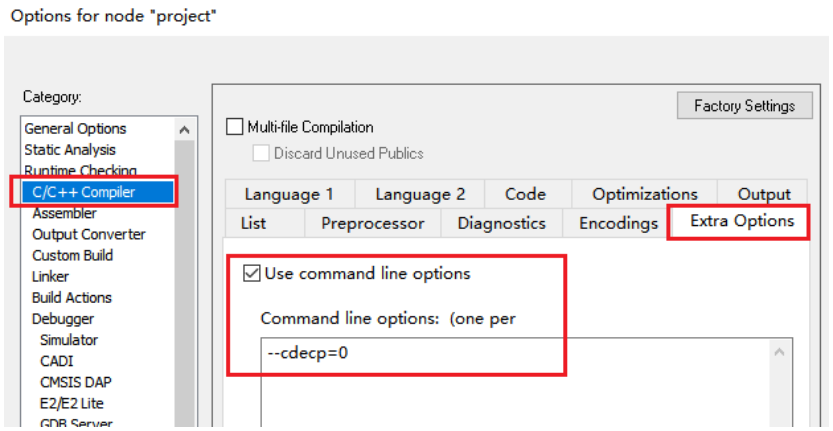


3.2.2 IAR（ICC）

IAR EW for Arm support is compilation support.

The compilation support can be enabled by checking Use command line options in Extra Options under C/C++ Compiler, and then adding “--cdecp=0” in the command line window.

Figure 2 ICC Enables CDE Compilation



3.3 Software Design Examples

This section takes the MDK-ARM environment as an example to introduce how to use Zidian to accelerate computation.

3.3.1 Combining driverlib with Zidian to accelerate computation

The specific steps for using Zidian for calculation in the SDK-driverlib routine are as follows:

- Enable compiler support
- Add the "__ZIDIAN_FCAU__" macro under the C/C++ tab in the engineering configuration to redirect the trigonometric functions supported by Zidian.
- Include "zidian_math.h" in the source code that requires Zidian to participate in calculation.

This application is described in detail with reference to the SDK driverlib routine zidian_ex1_math.

The routine uses Zidian to calculate and test sqrt, sin, cos, atan, and atan2. The specific code is as follows:

Since these functions have been packaged in the "zidian_math.h" file, to use Zidian to calculate these functions, you only need to quote "zidian_math.h" directly in the header file.

```
#include "zidian_math.h"

//
// Main
//
void example_main(void)
{
    volatile float x, y;
    uint16_t i;

    //
    // Test1 sqrtf
    //
    x = 0.81f;

    GET_DWT_CYCLE_COUNT(dwtCycleCounts[0],trace1Result[0] = sqrtf(x));

    for(i = 0; i < COUNT; i++)
    {
        trace1Result[i] = sqrtf(x);
    };

    //
    // Test2 sinf
    //
    x = PI / 2;
    GET_DWT_CYCLE_COUNT(dwtCycleCounts[1],trace2Result[1] = sinf(x));

    for(i = 0; i < COUNT; i++)          for(i = 0; i < COUNT; i++)
```

```
{
    trace2Result[i] = sinf(x);
};

//
// Test3 cosf
//
x = PI / 4;
GET_DWT_CYCLE_COUNT(dwtCycleCounts[2], trace3Result[2] = cosf(x));

for(i = 0; i < COUNT; i++)          for(i = 0; i < COUNT; i++)
{
    trace3Result[i] = cosf(x);
};

//
// Test4 atanf
//
x = 5.0f;
GET_DWT_CYCLE_COUNT(dwtCycleCounts[3], trace4Result[3] = atanf(x));

for(i = 0; i < COUNT; i++)
{
    trace4Result[i] = atanf(x);
};

//
// Test5 atan2f
//
x = 5.0f;
y = 10.0f;
GET_DWT_CYCLE_COUNT(dwtCycleCounts[4], trace5Result[4] = atan2f(y, x));

for(i = 0; i < COUNT; i++)
{
    trace5Result[i] = atan2f(y, x);
};

//
// Loop
//
for(;;)
```

```
    {
    }
}
```

3.3.2 Combining Math library with Zidian to Accelerate Computation

To use the Zidian acceleration function in the math library to accelerate computation, the specific steps are as follows:

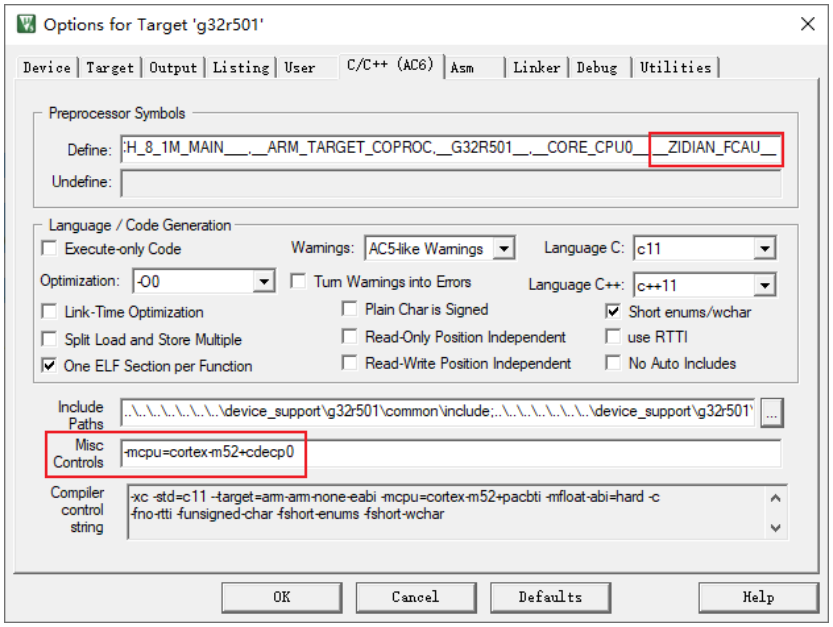
- Enable compiler support
- Add the "__ZIDIAN_FCAU__" macro under the C/C++ tab in the engineering configuration to redirect the mathematical library functions supported by Zidian.

Take the FPUfastRTS routine as an example, and use Zidian for computation acceleration.

Since the sin and cos functions are packaged in the FPUfastRTS library, if the "__ZIDIAN_FCAU__" macro is not added in the engineering configuration, calling such functions will use the functions in the FPUfastRTS library.

To use Zidian to accelerate computation, it is required to add the corresponding macro in the option under the C/C++ tab of the corresponding engineering configuration, and then call the objective function.

Figure 3 Using Zidian for Acceleration in Math Library



After enabling compiler support and adding the __ZIDIAN_FCAU__ macro, directly call the corresponding function. The specific code is as follows:

```
    in.f32    = test_input[i];
```

```
// Run the calculation function and measure the DWT cycle count simultaneously
GET_DWT_CYCLE_COUNT(dwtCycleCounts[0], out.f32    = atanf(in.f32));


test_output[i] = out.f32;
```

4 Revision

Table 5 Document Revision History

Date	Version	Change History
January, 2025	1.0	New

Statement

This document is formulated and published by Geehy Semiconductor Co., Ltd. (hereinafter referred to as “Geehy”). The contents in this document are protected by laws and regulations of trademark, copyright and software copyright. Geehy reserves the right to make corrections and modifications to this document at any time. Read this document carefully before using Geehy products. Once you use the Geehy product, it means that you (hereinafter referred to as the “users”) have known and accepted all the contents of this document. Users shall use the Geehy product in accordance with relevant laws and regulations and the requirements of this document.

1. Ownership

This document can only be used in connection with the corresponding chip products or software products provided by Geehy. Without the prior permission of Geehy, no unit or individual may copy, transcribe, modify, edit or disseminate all or part of the contents of this document for any reason or in any form.

The “极海” or “Geehy” words or graphics with “®” or “™” in this document are trademarks of Geehy. Other product or service names displayed on Geehy products are the property of their respective owners.

2. No Intellectual Property License

Geehy owns all rights, ownership and intellectual property rights involved in this document.

Geehy shall not be deemed to grant the license or right of any intellectual property to users explicitly or implicitly due to the sale or distribution of Geehy products or this document.

If any third party’s products, services or intellectual property are involved in this document, it shall not be deemed that Geehy authorizes users to use the aforesaid third party’s products, services or intellectual property. Any information regarding the application of the product, Geehy hereby disclaims any and all warranties and liabilities of any kind, including without limitation warranties of non-infringement of intellectual property rights of any third party, unless otherwise agreed in sales order or sales contract.

3. Version Update

Users can obtain the latest document of the corresponding models when ordering Geehy products.

If the contents in this document are inconsistent with Geehy products, the agreement in the sales order or the sales contract shall prevail.

4. Information Reliability

The relevant data in this document are obtained from batch test by Geehy Laboratory or cooperative third-party testing organization. However, clerical errors in correction or errors caused by differences in testing environment may occur inevitably. Therefore, users should understand that Geehy does not bear any responsibility for such errors that may occur in this document. The relevant data in this document are only used to guide users as performance parameter reference and do not constitute Geehy's guarantee for any product performance.

Users shall select appropriate Geehy products according to their own needs, and effectively verify and test the applicability of Geehy products to confirm that Geehy products meet their own needs, corresponding standards, safety or other reliability requirements. If losses are caused to users due to user's failure to fully verify and test Geehy products, Geehy will not bear any responsibility.

5. Legality

USERS SHALL ABIDE BY ALL APPLICABLE LOCAL LAWS AND REGULATIONS WHEN USING THIS DOCUMENT AND THE MATCHING GEEHY PRODUCTS. USERS SHALL UNDERSTAND THAT THE PRODUCTS MAY BE RESTRICTED BY THE EXPORT, RE-EXPORT OR OTHER LAWS OF THE COUNTRIES OF THE PRODUCTS SUPPLIERS, GEEHY, GEEHY DISTRIBUTORS AND USERS. USERS (ON BEHALF OR ITSELF, SUBSIDIARIES AND AFFILIATED ENTERPRISES) SHALL AGREE AND PROMISE TO ABIDE BY ALL APPLICABLE LAWS AND REGULATIONS ON THE EXPORT AND RE-EXPORT OF GEEHY PRODUCTS AND/OR TECHNOLOGIES AND DIRECT PRODUCTS.

6. Disclaimer of Warranty

THIS DOCUMENT IS PROVIDED BY GEEHY "AS IS" AND THERE IS NO WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE, TO THE EXTENT PERMITTED BY APPLICABLE LAW.

GEEHY'S PRODUCTS ARE NOT DESIGNED, AUTHORIZED, OR WARRANTED FOR USE AS CRITICAL COMPONENTS IN MILITARY, LIFE-SUPPORT, POLLUTION CONTROL, OR HAZARDOUS SUBSTANCES MANAGEMENT SYSTEMS, NOR WHERE FAILURE COULD RESULT IN INJURY, DEATH, PROPERTY OR ENVIRONMENTAL DAMAGE.

IF THE PRODUCT IS NOT LABELED AS "AUTOMOTIVE GRADE," IT SHOULD NOT BE CONSIDERED SUITABLE FOR AUTOMOTIVE APPLICATIONS. GEEHY ASSUMES NO LIABILITY FOR THE USE BEYOND ITS SPECIFICATIONS OR GUIDELINES.

THE USER SHOULD ENSURE THAT THE APPLICATION OF THE PRODUCTS COMPLIES WITH ALL RELEVANT STANDARDS, INCLUDING BUT NOT LIMITED TO SAFETY, INFORMATION SECURITY, AND ENVIRONMENTAL REQUIREMENTS. THE USER ASSUMES FULL RESPONSIBILITY FOR THE SELECTION AND USE OF GEEHY PRODUCTS. GEEHY WILL BEAR NO RESPONSIBILITY FOR ANY DISPUTES ARISING FROM THE SUBSEQUENT DESIGN OR USE BY USERS.

7. Limitation of Liability

IN NO EVENT, UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL GEEHY OR ANY OTHER PARTY WHO PROVIDES THE DOCUMENT AND PRODUCTS "AS IS", BE LIABLE FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, DIRECT, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE DOCUMENT AND PRODUCTS (INCLUDING BUT NOT LIMITED TO LOSSES OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY USERS OR THIRD PARTIES). THIS COVERS POTENTIAL DAMAGES TO PERSONAL SAFETY, PROPERTY, OR THE ENVIRONMENT, FOR WHICH GEEHY WILL NOT BE RESPONSIBLE.

8. Scope of Application

The information in this document replaces the information provided in all previous versions of the document.