GigaDevice Semiconductor Inc.

Dhryston Porting Guide Based on GD32 MCU

Application Notes

AN164

Version 1.0

(Dec 2025)

Table of Contents

List of Figures

List of Tables

1. Foreword

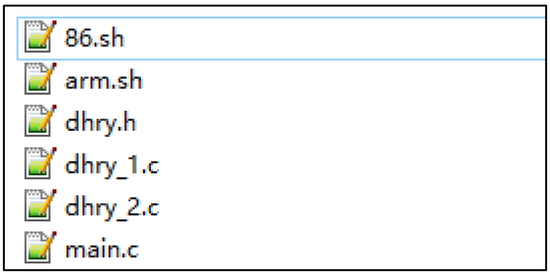
Dhrystone, one of the most common benchmark test programs, measures MCU computing capabilities. Its main purpose is to test the performance of integer operations and logical operations of MCUs.

The testing principle of Dhrystone involves measuring how many times an MCU executes the Dhrystone program within a given unit of time. The test results are expressed in DMIPS/MHz. DMIPS, short for abbreviation of Dhrystone Million Instructions Per Second, represents the number of millions of machine language instructions processed per second.

2. Dhrystone source code

The Dhrystone source code was downloaded from the Internet; we are currently testing the Version 2.1 C language version. After decompression, the source files appear as shown in ***Figure 2-1. Dhrystone source file***.

Figure 2-1. Dhrystone source file



86.sh
arm.sh
dhry.h
dhry_1.c
dhry_2.c
main.c

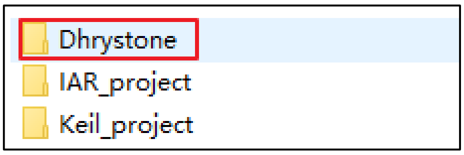
3. Dhrystone porting

3.1. Project configuration

This AN uses GD32L23x and keil5 as examples to describe Dhrystone porting and notes. Because GD32L23x is an ARM Cortex-M23 core, the files for testing Dhrystone are dhry.h, dhry_1.c, and dhry_2.c.

Create a Dhrystone folder under the project path, and copy the three files dhry.h, dhry_1.c, and dhry_2.c to this folder, as shown in ***Figure 3-1. Project directory structure*** and ***Figure 3-2. Dhrystone files***:

Figure 3-1. Project directory structure



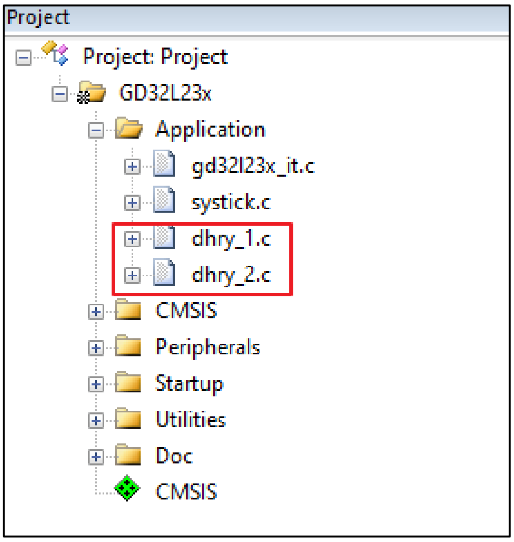
Dhrystone
IAR_project
Keil_project

Figure 3-2. Dhrystone files

dhry.h
dhry_1.c
dhry_2.c

Open the project, and add dhry_1.c and dhry_2.c to it, as shown in ***Figure 3-3. Dhrystone project structure***:

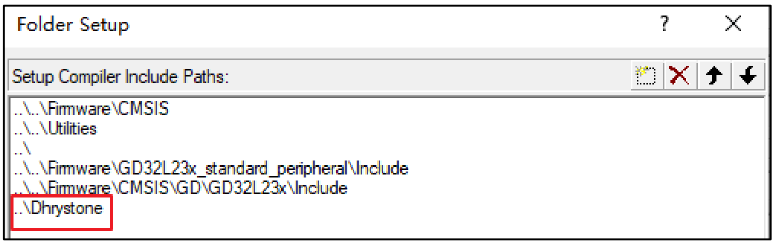
Figure 3-3. Dhrystone project structure



```
Project
⊟ Project: Project
    ⊟ GD32L23x
        ⊟ Application
            ⊞ gd32l23x_it.c
            ⊞ systick.c
            ⊞ dhry_1.c
            ⊞ dhry_2.c
        ⊞ CMSIS
        ⊞ Peripherals
        ⊞ Startup
        ⊞ Utilities
        ⊞ Doc
          CMSIS
```

Add the file inclusion path, as shown in ***Figure 3-4. Configuration of Dhrystone project file path***:

Figure 3-4. Configuration of Dhrystone project file path



```
..\..\Firmware\CMSIS
..\..\Utilities
..\
..\..\Firmware\GD32L23x_standard_peripheral\Include
..\..\Firmware\CMSIS\GD\GD32L23x\Include
..\Dhrystone
```

3.2. Code modification

The MCU requires timing and printing essential information when running the Dhrystone test code. Here, TIMER1, TIMER2, and USART1 are selected (for reference, select them as needed). Configure TIMER, USART, and the corresponding GPIO codes, as shown in ***Table 3-1. Clock configuration***, ***Table 3-2. USART and GPIO configuration***, and ***Table 3-3. TIMER configuration***:

Table 3-1. Clock configuration

```
void rcu_configuration(void)
{
    rcu_periph_clock_enable(RCU_USART1);
    rcu_periph_clock_enable(RCU_GPIOA);
    rcu_periph_clock_enable(RCU_TIMER1);
    rcu_periph_clock_enable(RCU_TIMER2);
}
```

Table 3-2. USART and GPIO configuration

```
void usart_config(void)
{
    /* connect port to USARTx_Tx */
    gpio_af_set(GPIOA, GPIO_AF_7, GPIO_PIN_2);

    /* connect port to USARTx_Rx */
    gpio_af_set(GPIOA, GPIO_AF_7, GPIO_PIN_3);

    /* configure USART Tx as alternate function push-pull */
    gpio_mode_set(GPIOA, GPIO_MODE_AF, GPIO_PUPD_PULLUP, GPIO_PIN_2);
    gpio_output_options_set(GPIOA, GPIO_OTYPE_PP, GPIO_OSPEED_10MHZ, GPIO_PIN_2);

    /* configure USART Rx as alternate function push-pull */
    gpio_mode_set(GPIOA, GPIO_MODE_AF, GPIO_PUPD_PULLUP, GPIO_PIN_3);
    gpio_output_options_set(GPIOA, GPIO_OTYPE_PP, GPIO_OSPEED_10MHZ, GPIO_PIN_3);

    /* USART configure */
    usart_deinit(USART1);
    usart_baudrate_set(USART1, 115200U);
    usart_receive_config(USART1, USART_RECEIVE_ENABLE);
    usart_transmit_config(USART1, USART_TRANSMIT_ENABLE);
    usart_enable(USART1);
}
```

Table 3-3. TIMER configuration

```
void timer_config(void)
{
    timer_parameter_struct timer_initpara;

    /* deinit TIMER */
    timer_deinit(TIMER1);
    timer_deinit(TIMER2);
    /* initialize TIMER init parameter struct */
    timer_struct_para_init(&timer_initpara);
    /* TIMER1 configuration */
    timer_initpara.prescaler          = ((clk/1000000)*2 -1);
    timer_initpara.alignedmode         = TIMER_COUNTER_EDGE;
    timer_initpara.counterdirection = TIMER_COUNTER_UP;
    timer_initpara.period              = 9999;
    timer_initpara.clockdivision      = TIMER_CKDIV_DIV1;
    timer_init(TIMER1, &timer_initpara);

    /* TIMER2 configuration */
    timer_initpara.prescaler          = 0;
    timer_initpara.alignedmode         = TIMER_COUNTER_EDGE;
    timer_initpara.counterdirection = TIMER_COUNTER_UP;
    timer_initpara.period              = 9999;
    timer_initpara.clockdivision      = TIMER_CKDIV_DIV1;
    timer_init(TIMER2, &timer_initpara);

    timer_master_slave_mode_config(TIMER1, TIMER_MASTER_SLAVE_MODE_ENABLE);
    timer_master_output_trigger_source_select(TIMER1, TIMER_TRI_OUT_SRC_UPDATE);

    timer_master_slave_mode_config(TIMER2, TIMER_MASTER_SLAVE_MODE_ENABLE);
    timer_slave_mode_select(TIMER2, TIMER_SLAVE_MODE_EXTERNAL0);
    timer_input_trigger_source_select(TIMER2, TIMER_SMCFG_TRGSEL_ITI0);

    /* enable TIMER */
    timer_enable(TIMER1);
    timer_enable(TIMER2);
}
```

The main function in the dhry_1.c file needs to be modified, as shown in ***Table 3-4. Modification of main function***:

Table 3-4. Modification of main function

```
int main (void)
/*****/

  /* main program, corresponds to procedures        */
  /* Main and Proc_0 in the Ada version             */
{
          One_Fifty       Int_1_Loc;
    REG   One_Fifty       Int_2_Loc;
          One_Fifty       Int_3_Loc;
    REG   char            Ch_Index;
          Enumeration     Enum_Loc;
          Str_30          Str_1_Loc;
          Str_30          Str_2_Loc;
    REG   int             Run_Index;
    REG   int             Number_Of_Runs;

    clk = rcu_clock_freq_get(CK_APB1);

    Next_Ptr_Glob = (Rec_Pointer) malloc (sizeof (Rec_Type));
    Ptr_Glob = (Rec_Pointer) malloc (sizeof (Rec_Type));

    Ptr_Glob->Ptr_Comp                    = Next_Ptr_Glob;
    Ptr_Glob->Discr                       = Ident_1;
    Ptr_Glob->variant.var_1.Enum_Comp     = Ident_3;
    Ptr_Glob->variant.var_1.Int_Comp      = 20;
    strcpy (Ptr_Glob->variant.var_1.Str_Comp,
          "DHRYSTONE PROGRAM, SOME STRING");
    strcpy (Str_1_Loc, "DHRYSTONE PROGRAM, 1'ST STRING");

    Arr_2_Glob [8][7] = 10;

    /* Was missing in published program. Without this statement,    */
    /* Arr_2_Glob [8][7] would have an undefined value.             */
    /* Warning: With 16-Bit processors and Number_Of_Runs > 32000,  */
    /* overflow may occur for this array element.                   */

    rcu_configuration();
    usart_config();

    printf ("Dhrystone Benchmark, Version 2.1 (Language: C)\n\r");
```

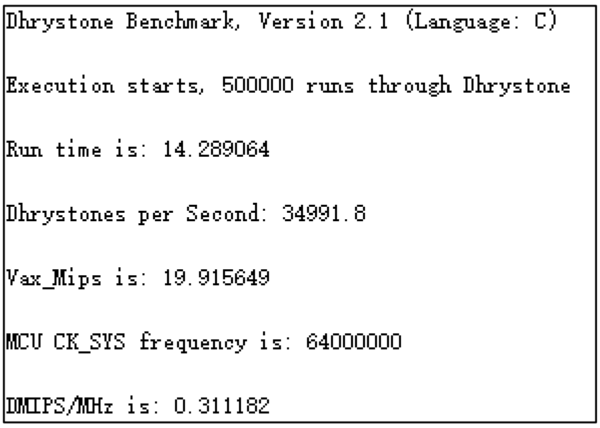
```
Number_Of_Runs = 1000000;
printf ("Execution starts, %d runs through Dhrystone\n\r", Number_Of_Runs);

timer_config();

Begin_Time = timer_counter_read(TIMER2)*10000 + timer_counter_read(TIMER1);
for (Run_Index = 1; Run_Index <= Number_Of_Runs; ++Run_Index)
{

    Proc_5();
    Proc_4();
    /* Ch_1_Glob == 'A', Ch_2_Glob == 'B', Bool_Glob == true */
    Int_1_Loc = 2;
    Int_2_Loc = 3;
    strcpy (Str_2_Loc, "DHRYSTONE PROGRAM, 2'ND STRING");
    Enum_Loc = Ident_2;
    Bool_Glob = ! Func_2 (Str_1_Loc, Str_2_Loc);
    /* Bool_Glob == 1 */
    while (Int_1_Loc < Int_2_Loc)    /* loop body executed once */
    {
        Int_3_Loc = 5 * Int_1_Loc - Int_2_Loc;
        /* Int_3_Loc == 7 */
        Proc_7 (Int_1_Loc, Int_2_Loc, &Int_3_Loc);
        /* Int_3_Loc == 7 */
        Int_1_Loc += 1;
    } /* while */
    /* Int_1_Loc == 3, Int_2_Loc == 3, Int_3_Loc == 7 */
    Proc_8 (Arr_1_Glob, Arr_2_Glob, Int_1_Loc, Int_3_Loc);
    /* Int_Glob == 5 */
    Proc_1 (Ptr_Glob);
    for (Ch_Index = 'A'; Ch_Index <= Ch_2_Glob; ++Ch_Index)
    /* loop body executed twice */
    {
        if (Enum_Loc == Func_1 (Ch_Index, 'C'))
        /* then, not executed */
        {
            Proc_6 (Ident_1, &Enum_Loc);
            strcpy (Str_2_Loc, "DHRYSTONE PROGRAM, 3'RD STRING");
            Int_2_Loc = Run_Index;
            Int_Glob = Run_Index;
        }
    }
```

```
        /* Int_1_Loc == 3, Int_2_Loc == 3, Int_3_Loc == 7 */
        Int_2_Loc = Int_2_Loc * Int_1_Loc;
        Int_1_Loc = Int_2_Loc / Int_3_Loc;
        Int_2_Loc = 7 * (Int_2_Loc - Int_3_Loc) - Int_1_Loc;
        /* Int_1_Loc == 1, Int_2_Loc == 13, Int_3_Loc == 7 */
        Proc_2 (&Int_1_Loc);
        /* Int_1_Loc == 5 */

    } /* loop "for Run_Index" */

    /**************/
    /* Stop timer */
    /**************/
    timer_disable(TIMER1);
    timer_disable(TIMER2);
    End_Time = timer_counter_read(TIMER2)*10000 + timer_counter_read(TIMER1);
    User_Time = End_Time - Begin_Time;

    Dhrystones_Per_Second = (double) Number_Of_Runs / (User_Time / 1000000);
    Vax_Mips = Dhrystones_Per_Second / 1757.0;

    printf ("Run time is: %6.6f \n\r", User_Time/1000000);
    printf ("Dhrystones per Second: %6.1f \n\r", Dhrystones_Per_Second);
    printf ("Vax_Mips is: %6.1f \n", Vax_Mips);
    printf ("\n");
    printf("MCU CK_SYS frequency is: %d\n\r%d\n\r", rcu_clock_freq_get(CK_AHB));
    printf("DMIPS/MHz is: %f \n", (double)Vax_Mips / (rcu_clock_freq_get(CK_AHB)/1000000));
    while(1);
}
```

4. Test results

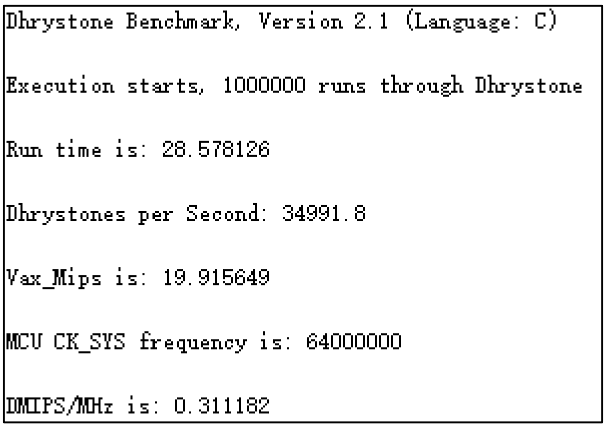
This document takes the code running in Flash as an example. Set Number_Of_Runs to 500000 and 10000000 respectively, where Dhrystones_Per_Second = number of runs/run time, Vax_Mips = Dhrystones_Per_Second/1757.0, and DMIPS/MHz = Vax_Mips/clock frequency. The corresponding test results are as shown in ***Figure 4-1. Dhrystone test 1*** and ***Figure 4-2. Dhrystone test 2***:

Figure 4-1. Dhrystone test 1



```
Dhrystone Benchmark, Version 2.1 (Language: C)

Execution starts, 500000 runs through Dhrystone

Run time is: 14.289064

Dhrystones per Second: 34991.8

Vax_Mips is: 19.915649

MCU CK_SYS frequency is: 64000000

DMIPS/MHz is: 0.311182
```

Figure 4-2. Dhrystone test 2



```
Dhrystone Benchmark, Version 2.1 (Language: C)

Execution starts, 1000000 runs through Dhrystone

Run time is: 28.578126

Dhrystones per Second: 34991.8

Vax_Mips is: 19.915649

MCU CK_SYS frequency is: 64000000

DMIPS/MHz is: 0.311182
```

5. Revision history

Table 5-1. Revision history

Revision No.	Description	Date
1.0	Initial release	Dec.05, 2025

Important Notice

This document is the property of GigaDevice Semiconductor Inc. and its subsidiaries (the "Company"). This document, including any product of the Company described in this document (the “Product”), is owned by the Company according to the laws of the People’s Republic of China and other applicable laws. The Company reserves all rights under such laws and no Intellectual Property Rights are transferred (either wholly or partially) or licensed by the Company (either expressly or impliedly) herein. The names and brands of third party referred thereto (if any) are the property of their respective owner and referred to for identification purposes only.

To the maximum extent permitted by applicable law, the Company makes no representations or warranties of any kind, express or implied, with regard to the merchantability and the fitness for a particular purpose of the Product, nor does the Company assume any liability arising out of the application or use of any Product. Any information provided in this document is provided only for reference purposes. It is the sole responsibility of the user of this document to determine whether the Product is suitable and fit for its applications and products planned, and properly design, program, and test the functionality and safety of its applications and products planned using the Product. The Product is designed, developed, and/or manufactured for ordinary business, industrial, personal, and/or household applications only, and the Product is not designed or intended for use in (i) safety critical applications such as weapons systems, nuclear facilities, atomic energy controller, combustion controller, aeronautic or aerospace applications, traffic signal instruments, pollution control or hazardous substance management; (ii) life-support systems, other medical equipment or systems (including life support equipment and surgical implants); (iii) automotive applications or environments, including but not limited to applications for active and passive safety of automobiles (regardless of front market or aftermarket), for example, EPS, braking, ADAS (camera/fusion), EMS, TCU, BMS, BSG, TPMS, Airbag, Suspension, DMS, ICMS, Domain, ESC, DCDC, e-clutch, advanced-lighting, etc.. Automobile herein means a vehicle propelled by a self-contained motor, engine or the like, such as, without limitation, cars, trucks, motorcycles, electric cars, and other transportation devices; and/or (iv) other uses where the failure of the device or the Product can reasonably be expected to result in personal injury, death, or severe property or environmental damage (collectively "Unintended Uses"). Customers shall take any and all actions to ensure the Product meets the applicable laws and regulations. The Company is not liable for, in whole or in part, and customers shall hereby release the Company as well as its suppliers and/or distributors from, any claim, damage, or other liability arising from or related to all Unintended Uses of the Product. Customers shall indemnify and hold the Company, and its officers, employees, subsidiaries, affiliates as well as its suppliers and/or distributors harmless from and against all claims, costs, damages, and other liabilities, including claims for personal injury or death, arising from or related to any Unintended Uses of the Product.

Information in this document is provided solely in connection with the Product. The Company reserves the right to make changes, corrections, modifications or improvements to this document and the Product described herein at any time without notice. The Company shall have no responsibility whatsoever for conflicts or incompatibilities arising from future changes to them. Information in this document supersedes and replaces information previously supplied in any prior versions of this document.