Gowin_EMPU_M1 Software Programming Reference Manual
<table>
<thead>
<tr>
<th>Date</th>
<th>Version</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>02/19/2019</td>
<td>1.0E</td>
<td>Initial version published.</td>
</tr>
<tr>
<td>07/18/2019</td>
<td>1.1E</td>
<td>MCU hardware design and software programming design support extended peripherals: CAN, Ethernet, SPI-Flash, RTC, DualTimer, TRNG, I2C, SPI, SD-Card.</td>
</tr>
<tr>
<td>08/18/2019</td>
<td>1.2E</td>
<td>• MCU hardware design and software programming design support extended peripheral: DDR3 Memory; Known issues of ITCM, DTCM Size and IDE fixed.</td>
</tr>
</tbody>
</table>
| 09/27/2019 | 1.3E    | • MCU hardware design and software programming design support read, write and erasure of SPI-Flash;  
• MCU software programming design supports a continuous multi-byte read and write of \( i^2C \);  
• Fixed known issues of address mapping of AHB2 and APB2 extended interface in MCU software programming design;  
• Fixed known issues of continuous read and write of DDR3 Memory in MCU software programming design. |
| 12/06/2019 | 1.4E    | • MCU hardware design and software programming design supports PSRAM;  
• MCU compiling software GMD V1.0 updated;  
• RTOS reference design updated;  
• Hardware and software reference design of AHB2 and APB2 extension bus interface added. |
| 03/09/2020 | 1.5E    | MCU software programming design supports the read and write of SD-Card.                                                                     |
| 06/12/2020 | 1.6E    | • MCU supports for external instruction memory;  
• MCU supports for external data memory;  
• Extension of 6 AHB bus interfaces;  
• Extension of 16 APB bus interfaces;  
• GPIO supports multiple interface types;  
• \( i^2C \) supports multiple interface types. |
## Contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Contents</td>
<td>i</td>
</tr>
<tr>
<td>List of Figures</td>
<td>iv</td>
</tr>
<tr>
<td>List of Tables</td>
<td>iv</td>
</tr>
<tr>
<td><strong>1 Software Programming Library</strong></td>
<td>1</td>
</tr>
<tr>
<td>1.1 Cortex-M1 Software Programming</td>
<td>1</td>
</tr>
<tr>
<td>1.2 Embedded Operating System Software</td>
<td>2</td>
</tr>
<tr>
<td><strong>2 Memory System</strong></td>
<td>3</td>
</tr>
<tr>
<td>2.1 Standard Peripherals Memory Mapping</td>
<td>3</td>
</tr>
<tr>
<td>2.2 MCU System Memory Mapping</td>
<td>4</td>
</tr>
<tr>
<td><strong>3 Interrupt Handling</strong></td>
<td>5</td>
</tr>
<tr>
<td><strong>4 Universal Asynchronous Receiver/Transmitter</strong></td>
<td>7</td>
</tr>
<tr>
<td>4.1 Features</td>
<td>7</td>
</tr>
<tr>
<td>4.2 Registers</td>
<td>8</td>
</tr>
<tr>
<td>4.3 Initialization</td>
<td>9</td>
</tr>
<tr>
<td>4.4 Drivers Usage</td>
<td>9</td>
</tr>
<tr>
<td>4.5 Reference Design</td>
<td>10</td>
</tr>
<tr>
<td><strong>5 Timer</strong></td>
<td>11</td>
</tr>
<tr>
<td>5.1 Features</td>
<td>11</td>
</tr>
<tr>
<td>5.2 Registers</td>
<td>11</td>
</tr>
<tr>
<td>5.3 Initialization</td>
<td>12</td>
</tr>
<tr>
<td>5.4 Drivers Usage</td>
<td>12</td>
</tr>
<tr>
<td>5.5 Reference Design</td>
<td>12</td>
</tr>
<tr>
<td><strong>6 Watchdog</strong></td>
<td>13</td>
</tr>
<tr>
<td>6.1 Features</td>
<td>13</td>
</tr>
<tr>
<td>6.2 Registers</td>
<td>14</td>
</tr>
<tr>
<td>6.3 Initialization</td>
<td>14</td>
</tr>
<tr>
<td>6.4 Drivers Usage</td>
<td>15</td>
</tr>
<tr>
<td>6.5 Reference Design</td>
<td>15</td>
</tr>
<tr>
<td><strong>7 GPIO</strong></td>
<td>16</td>
</tr>
<tr>
<td>7.1 Features</td>
<td>16</td>
</tr>
</tbody>
</table>
7.2 Registers ................................................................. 16
7.3 Initialization ............................................................. 20
7.4 Drivers Usage .......................................................... 21
7.5 Reference Design .................................................... 21

8 I²C ........................................................................ 22
8.1 Features ................................................................. 22
8.2 Registers ................................................................. 22
8.3 Drivers Usage .......................................................... 23
8.4 Reference Design .................................................... 23

9 SPI .......................................................................... 24
9.1 Features ................................................................. 24
9.2 Registers ................................................................. 24
9.3 Initialization ............................................................. 25
9.4 Drivers Usage .......................................................... 26
9.5 Reference Design .................................................... 26

10 Real-time Clock ......................................................... 27
10.1 Features ................................................................. 27
10.2 Registers ................................................................. 28
10.3 Drivers Usage .......................................................... 29
10.4 Reference Design .................................................... 29

11 True Random Number Generator .................................. 30
11.1 Features ................................................................. 30
11.2 Registers ................................................................. 30
11.3 Drivers Usage .......................................................... 33
11.4 Reference Design .................................................... 33

12 Dual Timer ................................................................. 34
12.1 Features ................................................................. 34
12.2 Registers ................................................................. 34
12.3 Drivers Usage .......................................................... 36
12.4 Reference Design .................................................... 36

13 SD Card ................................................................. 37
13.1 Features ................................................................. 37
13.2 Registers ................................................................. 37
13.3 Drivers Usage .......................................................... 40
13.4 Reference Design .................................................... 40

14 Controller Area Network ............................................... 41
14.1 Features ................................................................. 41
14.2 Registers ................................................................. 41
14.3 Driver Usage ................................................................. 45
14.4 Reference Design ....................................................... 47

15 Ethernet ........................................................................... 48
15.1 Features ......................................................................... 48
15.2 Registers ....................................................................... 49
15.3 Drivers Usage ............................................................. 51
15.4 Reference Design ........................................................ 51

16 DDR3 Memory .................................................................. 52
16.1 Features ......................................................................... 52
16.2 Registers ....................................................................... 53
16.3 Drivers Usage ............................................................. 53
16.4 Reference Design ........................................................ 53

17 SPI-Flash ........................................................................ 54
17.1 Features ......................................................................... 54
17.2 Registers ....................................................................... 54
17.3 Drivers Usage ............................................................. 60
17.4 Reference Design ........................................................ 60

18 PSRAM ............................................................................. 61
18.1 Features ......................................................................... 61
18.2 Registers ....................................................................... 62
18.3 Drivers Usage ............................................................. 63
18.4 Reference Design ........................................................ 63

19 Embedded Real-time Operating System ....................... 64
19.1 uC/OS-III ................................................................. 64
19.1.1 Features ..................................................................... 64
19.1.2 Operation System Version ........................................... 64
19.1.3 Operation System Configuration ................................. 64
19.1.4 Reference Design ........................................................ 65
19.2 FreeRTOS ................................................................. 65
19.2.1 Features ..................................................................... 65
19.2.2 Operation System Version ........................................... 65
19.2.3 Operation System Configuration ................................. 65
19.2.4 Reference Design ........................................................ 65
List of Figures

Figure 4-1 UART Buffering ............................................................................................................7
Figure 5-1 Timer Diagram .............................................................................................................11
Figure 6-1 Watch Dog Operation ...............................................................................................13
Figure 7-1 GPIO Block .................................................................................................................16
Figure 10-1 RTC Block .................................................................................................................28
List of Tables

Table 1-1 Cortex-M1 Core Software Programming ................................................................. 1
Table 2-1 Standard Peripheral Memory Mapping Definitions .................................................. 3
Table 2-2 Definition of Memory Mapping of Core System ....................................................... 4
Table 3-1 Definition of Interrupt Controller ........................................................................... 5
Table 4-1 Definition of UART Registers .................................................................................. 8
Table 4-2 UART Initialization Definition .............................................................................. 9
Table 4-3 Usage of UART Drivers ......................................................................................... 9
Table 5-1 Definition of Timer Registers ............................................................................... 11
Table 5-2 Timer Initialization Definition .............................................................................. 12
Table 5-3 Usage of Timer Drivers ....................................................................................... 12
Table 6-1 Definition of Watch Dog Registers ....................................................................... 14
Table 6-2 Initialization of Watch Dogs ................................................................................. 14
Table 6-3 Usage of Watch Dog Drivers ................................................................................. 15
Table 7-1 Definition of GPIO Registers ............................................................................... 16
Table 7-2 GPIO Initialization ............................................................................................... 20
Table 7-3 Usage of GPIO Drivers ......................................................................................... 21
Table 8-1 Definition of I\(^2\)C Master Registers ................................................................... 22
Table 8-2 Usage of I\(^2\)C Master Drivers ............................................................................ 23
Table 9-1 Definition of SPI Master Registers ....................................................................... 24
Table 9-2 SPI Master Initialization ....................................................................................... 25
Table 9-3 Usage of SPI Master Drivers ............................................................................... 26
Table 10-1 Definition of \(\dot{\text{I}}\)C Registers ........................................................................... 28
Table 10-2 Usage of RTC Drivers ....................................................................................... 29
Table 11-1 Definition of TRNG Registers ............................................................................. 30
Table 11-2 Usage of TRNG Driver ....................................................................................... 33
Table 12-1 Definition of DualTimer Registers ..................................................................... 34
Table 12-2 Usage of DualTimer Drivers .............................................................................. 36
Table 13-1 Definition of SD-Card Registers ......................................................................... 37
Table 13-2 Usage of SD-Card Drivers ................................................................................... 40
Table 14-1 Definition of CAN Registers ............................................................................... 41
Table 14-2 Usage of CAN Drivers ....................................................................................... 45
Table 15-1 Definition of Ethernet Registers ......................................................................... 49
Table 15-2 Usage of Ethernet Drivers .................................................................51
Table 16-1 Definition of DDR3 Registers ..........................................................53
Table 16-2 Usage of DDR3 Drivers .................................................................53
Table 17-1 Definition of SPI-Flash Registers ..................................................54
Table 17-2 Usage of SPI-Flash Drivers ............................................................60
Table 18-1 Definition of PSRAM Registers .....................................................62
Table 18-2 Usage of PSRAM Drivers ..............................................................63
1 Software Programming Library

Gowin_EMPU_M1 offers software programming library:
Gowin_EMPU_M1\src\c_lib

Note!

Two Gowin_EMPU_M1 software programming methods are supported:
- Cortex-M1 core software programming
- Embedded Operation System software programming

1.1 Cortex-M1 Core Software Programming

The Cortex-M1 software programming method provided in the Gowin_EMPU_M1 software programming library is shown in Table 1-1.

<table>
<thead>
<tr>
<th>File</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>startup_GOWIN_M1.s</td>
<td>Cortex-M1 core startup program</td>
</tr>
<tr>
<td>core_cm1.h</td>
<td>Definition of Cortex-M1 core register</td>
</tr>
<tr>
<td>GOWIN_M1.h</td>
<td>Definition of interrupt vector table, peripheral register, and address mapping</td>
</tr>
<tr>
<td>system_GOWIN_M1.c</td>
<td>Cortex-M1 core system initialization and system clock definition</td>
</tr>
<tr>
<td>GOWIN_M1_flash.ld</td>
<td>GMD Flash linker</td>
</tr>
<tr>
<td>GOWIN_M1_gpio.c</td>
<td>GPIO driving function definition</td>
</tr>
<tr>
<td>GOWIN_M1_can.c</td>
<td>CAN driving function definition</td>
</tr>
<tr>
<td>GOWIN_M1_ethernet.c</td>
<td>Ethernet driving function definition</td>
</tr>
<tr>
<td>GOWIN_M1_ddr3.c</td>
<td>DDR3 driving function definition</td>
</tr>
<tr>
<td>GOWIN_M1_psram.c</td>
<td>PSRAM driving function definition</td>
</tr>
<tr>
<td>GOWIN_M1_spi_flash.c</td>
<td>Driving function definition of SPI-Flash read, write and erasure</td>
</tr>
<tr>
<td>GOWIN_M1_timer.c</td>
<td>Timer driving function definition</td>
</tr>
<tr>
<td>GOWIN_M1_wdog.c</td>
<td>Definition of Watch Dog driving function</td>
</tr>
<tr>
<td>GOWIN_M1_uart.c</td>
<td>Definition of UART driving function</td>
</tr>
</tbody>
</table>
### 1.2 Embedded Operating System Software Programming

Gowin_EMPU_M1 supports the following two types of embedded RTOS software programming:

- uC/OS-III
- FreeRTOS

<table>
<thead>
<tr>
<th>File</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GOWIN_M1_rtc.c</td>
<td>RTC driving function definition</td>
</tr>
<tr>
<td>GOWIN_M1_trng.c</td>
<td>TRNG driving function definition</td>
</tr>
<tr>
<td>GOWIN_M1_dualtimer.c</td>
<td>DualTimer driving function definition</td>
</tr>
<tr>
<td>GOWIN_M1_i2c.c</td>
<td>I2C Master driving function definition</td>
</tr>
<tr>
<td>GOWIN_M1_spi.c</td>
<td>SPI Master driving function definition</td>
</tr>
<tr>
<td>GOWIN_M1_sdcard.c</td>
<td>SD-Card driving function definition</td>
</tr>
<tr>
<td>GOWIN_M1_m misc.c</td>
<td>Interrupt priority management and SysTick</td>
</tr>
<tr>
<td>retarget.c</td>
<td>Retargeting of UART0 printf function</td>
</tr>
<tr>
<td>malloc.c</td>
<td>Dynamic memory management</td>
</tr>
</tbody>
</table>
2 Memory System

2.1 Standard Peripherals Memory Mapping

The definition of memory mapping addresses of Gowin_EMPU_M1 standard peripherals are as shown in Table 2-1.

<table>
<thead>
<tr>
<th>Standard Peripheral</th>
<th>Type</th>
<th>Address Mapping</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ITCM</td>
<td>-</td>
<td>0x00000000</td>
<td>1KB, 2KB, 4KB, 8KB, 16KB, 32KB, 64KB, 128KB, 256KB</td>
</tr>
<tr>
<td>DTCM</td>
<td>-</td>
<td>0x20000000</td>
<td>1KB, 2KB, 4KB, 8KB, 16KB, 32KB, 64KB, 128KB, 256KB</td>
</tr>
<tr>
<td>External Instruction Memory</td>
<td>-</td>
<td>0x00000000</td>
<td>External instruction memory</td>
</tr>
<tr>
<td>External Data Memory</td>
<td>-</td>
<td>0x20100000</td>
<td>External data memory</td>
</tr>
<tr>
<td>TIMER0</td>
<td>TIMER_TypeDef</td>
<td>0x50000000</td>
<td>TIMER0</td>
</tr>
<tr>
<td>TIMER1</td>
<td>TIMER_TypeDef</td>
<td>0x50001000</td>
<td>Timer1</td>
</tr>
<tr>
<td>UART0</td>
<td>UART_TypeDef</td>
<td>0x50004000</td>
<td>UART0</td>
</tr>
<tr>
<td>UART1</td>
<td>UART_TypeDef</td>
<td>0x50005000</td>
<td>UART1</td>
</tr>
<tr>
<td>Watch Dog</td>
<td>WDOG_TypeDef</td>
<td>0x50008000</td>
<td>Watchdog</td>
</tr>
<tr>
<td>RTC</td>
<td>RTC_RegDef</td>
<td>0x50006000</td>
<td>Real-time clock</td>
</tr>
<tr>
<td>TRNG</td>
<td>TRNG_RegDef</td>
<td>0x5000F000</td>
<td>True Random Number Generator</td>
</tr>
<tr>
<td>DualTimer</td>
<td>DUALTIMER_RegDef</td>
<td>0x50002000</td>
<td>Dual Timer</td>
</tr>
<tr>
<td>SPI_FLASH</td>
<td>SPI_FLASH_RegDef</td>
<td>0x50003000</td>
<td>Serial Peripheral Interface Flash</td>
</tr>
<tr>
<td>I2C</td>
<td>I2C_TypeDef</td>
<td>0x5000A000</td>
<td>Internal Integrated Circuit Bus</td>
</tr>
<tr>
<td>SPI</td>
<td>SPI_TypeDef</td>
<td>0x5000B000</td>
<td>SPI</td>
</tr>
<tr>
<td>SD-Card</td>
<td>SDCard_TypeDef</td>
<td>0x50009000</td>
<td>SD</td>
</tr>
<tr>
<td>GPIO0</td>
<td>GPIO_TypeDef</td>
<td>0x40000000</td>
<td>GPIO port</td>
</tr>
<tr>
<td>CAN</td>
<td>CAN_RegDef</td>
<td>0x45000000</td>
<td>Controller Area Network</td>
</tr>
</tbody>
</table>
### Standard Peripheral | Type | Address Mapping | Description
--- | --- | --- | ---
Ethernet | ETH_RegDef | 0x46000000 | Ethernet
DDR3 | DDR3_RegDef | 0x88000000 | DDR3 Memory
PSRAM | PSRAM_TypeDef | 0x82000000 | Pseudo Static Random Access Memory
APB Master [1] | - | 0x60000000 | Extension of APB Master [1]
APB Master [8] | - | 0x60700000 | Extension of APB Master [8]
APB Master [9] | - | 0x60800000 | Extension of APB Master [9]
APB Master [10] | - | 0x60900000 | Extension of APB Master [10]
APB Master [12] | - | 0x60B00000 | Extension of APB Master [12]
APB Master [14] | - | 0x60D00000 | Extension of APB Master [14]
APB Master [16] | - | 0x60F00000 | Extension of APB Master [16]
AHB Master [1] | - | 0x80000000 | Extension of AHB Master [1]

## 2.2 MCU System Memory Mapping

The definition of Cortex-M1 memory mapping of core system is as shown in Table 2-2.

### Table 2-2 Definition of Memory Mapping of Core System

<table>
<thead>
<tr>
<th>System Control</th>
<th>Type</th>
<th>Address Mapping</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SysTick</td>
<td>SysTick_Type</td>
<td>0xE000E010</td>
<td>SysTick configuration struct</td>
</tr>
<tr>
<td>NVIC</td>
<td>NVIC_BASE</td>
<td>0xE000E100</td>
<td>NVIC configuration struct</td>
</tr>
<tr>
<td>SCnSCB</td>
<td>SCnSCB_Type</td>
<td>0xE000E000</td>
<td>System control register not in SCB</td>
</tr>
<tr>
<td>SCB</td>
<td>SCB_Type</td>
<td>0xE000ED00</td>
<td>SCB configuration struct</td>
</tr>
</tbody>
</table>
Interrupt Handling

Gowin_EMPU_M1 nested vectored interrupt controller (NVIC) includes the following features:

- It can provide up to 32 interrupt processing signals that can be used; 1, 8, 16, or 32 external interrupt processing signals can be configured;
- Supports programmable priorities of 0-3.

The definition of Cortex-M1 interrupt controller is as shown in Table 3-1.

<table>
<thead>
<tr>
<th>Address</th>
<th>Interrupt</th>
<th>Number</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x00000000</td>
<td>_StackTop</td>
<td>-</td>
<td>Top of Stack</td>
</tr>
<tr>
<td>0x00000004</td>
<td>Reset_Handler</td>
<td>-</td>
<td>Reset Handler</td>
</tr>
<tr>
<td>0x00000008</td>
<td>NMI_Handler</td>
<td>-14</td>
<td>NMI Handler</td>
</tr>
<tr>
<td>0x0000000C</td>
<td>HardFault_Handler</td>
<td>-13</td>
<td>Hard Fault Handler</td>
</tr>
<tr>
<td>0x00000010</td>
<td>0</td>
<td>-</td>
<td>Reserved</td>
</tr>
<tr>
<td>0x00000014</td>
<td>0</td>
<td>-</td>
<td>Reserved</td>
</tr>
<tr>
<td>0x00000018</td>
<td>0</td>
<td>-</td>
<td>Reserved</td>
</tr>
<tr>
<td>0x0000001C</td>
<td>0</td>
<td>-</td>
<td>Reserved</td>
</tr>
<tr>
<td>0x00000020</td>
<td>0</td>
<td>-</td>
<td>Reserved</td>
</tr>
<tr>
<td>0x00000024</td>
<td>0</td>
<td>-</td>
<td>Reserved</td>
</tr>
<tr>
<td>0x00000028</td>
<td>0</td>
<td>-</td>
<td>Reserved</td>
</tr>
<tr>
<td>0x0000002C</td>
<td>SVC_Handler</td>
<td>-5</td>
<td>SVCall Handler</td>
</tr>
<tr>
<td>0x00000030</td>
<td>0</td>
<td>-</td>
<td>Reserved</td>
</tr>
<tr>
<td>0x00000034</td>
<td>0</td>
<td>-</td>
<td>Reserved</td>
</tr>
<tr>
<td>0x00000038</td>
<td>PendSV_Handler</td>
<td>-2</td>
<td>PendSV Handler</td>
</tr>
<tr>
<td>0x0000003C</td>
<td>SysTick_Handler</td>
<td>-1</td>
<td>SysTick Handler</td>
</tr>
<tr>
<td>0x00000040</td>
<td>UART0_Handler</td>
<td>0</td>
<td>16+ 0: UART 0 RX and TX Handler</td>
</tr>
<tr>
<td>0x00000044</td>
<td>UART1_Handler</td>
<td>1</td>
<td>16+ 1: UART 1 RX and TX Handler</td>
</tr>
<tr>
<td>0x00000048</td>
<td>TIMER0_Handler</td>
<td>2</td>
<td>16+ 2: TIMR 0 handler</td>
</tr>
<tr>
<td>0x0000004C</td>
<td>TIMER1_Handler</td>
<td>3</td>
<td>16+ 3: TIMER 1 handler</td>
</tr>
</tbody>
</table>
## 3 Interrupt Handling

<table>
<thead>
<tr>
<th>Address</th>
<th>Interrupt</th>
<th>Number</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x00000050</td>
<td>GPIO0_Handler</td>
<td>4</td>
<td>16+ 4: GPIO Port 0 Combined Handler</td>
</tr>
<tr>
<td>0x00000054</td>
<td>UARTOVF_Handler</td>
<td>5</td>
<td>16+ 5: UART 0,1 Overflow Handler</td>
</tr>
<tr>
<td>0x00000058</td>
<td>RTC_Handler</td>
<td>6</td>
<td>16+ 6: RTC Handler</td>
</tr>
<tr>
<td>0x0000005C</td>
<td>I2C_Handler</td>
<td>7</td>
<td>16+ 7: I2C Handler</td>
</tr>
<tr>
<td>0x00000060</td>
<td>CAN_Handler</td>
<td>8</td>
<td>16+ 8: CAN Handler</td>
</tr>
<tr>
<td>0x00000064</td>
<td>ETH_Handler</td>
<td>9</td>
<td>16+ 9: ETH Handler</td>
</tr>
<tr>
<td>0x00000068</td>
<td>Interrupt10_Handler</td>
<td>10</td>
<td>16+10: Interrupt 10 Handler</td>
</tr>
<tr>
<td>0x0000006C</td>
<td>DTimer_Handler</td>
<td>11</td>
<td>16+11: DualTimer Handler</td>
</tr>
<tr>
<td>0x00000070</td>
<td>TRNG_Handler</td>
<td>12</td>
<td>16+12: TRNG Handler</td>
</tr>
<tr>
<td>0x00000074</td>
<td>Interrupt13_Handler</td>
<td>13</td>
<td>16+13: Interrupt 13 Handler</td>
</tr>
<tr>
<td>0x00000078</td>
<td>Interrupt14_Handler</td>
<td>14</td>
<td>16+14: Interrupt 14 Handler</td>
</tr>
<tr>
<td>0x0000007C</td>
<td>Interrupt15_Handler</td>
<td>15</td>
<td>16+15: Interrupt 15 Handler</td>
</tr>
<tr>
<td>0x00000080</td>
<td>GPIO0_0_Handler</td>
<td>16</td>
<td>16+16: GPIO0_0 Handler</td>
</tr>
<tr>
<td>0x00000084</td>
<td>GPIO0_1_Handler</td>
<td>17</td>
<td>16+17: GPIO0_1 Handler</td>
</tr>
<tr>
<td>0x00000088</td>
<td>GPIO0_2_Handler</td>
<td>18</td>
<td>16+18: GPIO0_2 Handler</td>
</tr>
<tr>
<td>0x0000008C</td>
<td>GPIO0_3_Handler</td>
<td>19</td>
<td>16+19: GPIO0_3 Handler</td>
</tr>
<tr>
<td>0x00000090</td>
<td>GPIO0_4_Handler</td>
<td>20</td>
<td>16+20: GPIO0_4 Handler</td>
</tr>
<tr>
<td>0x00000094</td>
<td>GPIO0_5_Handler</td>
<td>21</td>
<td>16+21: GPIO0_5 Handler</td>
</tr>
<tr>
<td>0x00000098</td>
<td>GPIO0_6_Handler</td>
<td>22</td>
<td>16+22: GPIO0_6 Handler</td>
</tr>
<tr>
<td>0x0000009C</td>
<td>GPIO0_7_Handler</td>
<td>23</td>
<td>16+23: GPIO0_7 Handler</td>
</tr>
<tr>
<td>0x000000A0</td>
<td>GPIO0_8_Handler</td>
<td>24</td>
<td>16+24: GPIO0_8 Handler</td>
</tr>
<tr>
<td>0x000000A4</td>
<td>GPIO0_9_Handler</td>
<td>25</td>
<td>16+25: GPIO0_9 Handler</td>
</tr>
<tr>
<td>0x000000A8</td>
<td>GPIO0_10_Handler</td>
<td>26</td>
<td>16+26: GPIO0_10 Handler</td>
</tr>
<tr>
<td>0x000000AC</td>
<td>GPIO0_11_Handler</td>
<td>27</td>
<td>16+27: GPIO0_11 Handler</td>
</tr>
<tr>
<td>0x000000B0</td>
<td>GPIO0_12_Handler</td>
<td>28</td>
<td>16+28: GPIO0_12 Handler</td>
</tr>
<tr>
<td>0x000000B4</td>
<td>GPIO0_13_Handler</td>
<td>29</td>
<td>16+29: GPIO0_13 Handler</td>
</tr>
<tr>
<td>0x000000B8</td>
<td>GPIO0_14_Handler</td>
<td>30</td>
<td>16+30: GPIO0_14 Handler</td>
</tr>
<tr>
<td>0x000000BC</td>
<td>GPIO0_15_Handler</td>
<td>31</td>
<td>16+31: GPIO0_15 Handler</td>
</tr>
</tbody>
</table>
4 Universal Asynchronous Receiver/Transmitter

4.1 Features

Gowin_EMPU_M1 includes two UARTs accessed by APB bus:

- The max. baud rate is 921.6Kbit/s
- No parity bit
- 8-bit data bit
- 1-bit stop bit

UART buffering is as shown in Figure 4-1.

Figure 4-1 UART Buffering

The UART supports the High Speed Test Mode (HSTM). When the register CTRL [6] is set to 1, the serial data is transmitted one bit per cycle, and the text information can be transmitted in a short time.

The baud rate divider register should be set when using UART. For
example, if the APB1 bus frequency is running at 12MHz and the baud rate is required to be 9600, the baud rate divider register can be set to 12000000/9600=1250.

4.2 Registers

The definition of UART registers is as shown in Table 4-1.

<table>
<thead>
<tr>
<th>Register Name</th>
<th>Address Offset</th>
<th>Type</th>
<th>Width</th>
<th>Initial Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DATA</td>
<td>0x000</td>
<td>RW</td>
<td>8</td>
<td>0x--</td>
<td>[7:0] Data Value</td>
</tr>
<tr>
<td>STATE</td>
<td>0x004</td>
<td>RW</td>
<td>4</td>
<td>0x0</td>
<td>[3] RX buffer overrun,write 1 to clear</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>[2] TX buffer overrun,write 1 to clear</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>[1] RX buffer full, read-only</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>[0] TX buffer full, read-only</td>
</tr>
<tr>
<td>CTRL</td>
<td>0x008</td>
<td>RW</td>
<td>7</td>
<td>0x00</td>
<td>[6] High speed test mode for TX only</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>[5] RX overrun interrupt enable</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>[4] TX overrun interrupt enable</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>[3] RX interrupt enable</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>[2] TX interrupt enable</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>[1] RX enable</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>[0] TX enable</td>
</tr>
<tr>
<td>INTSTATUS/</td>
<td>0x00C</td>
<td>RW</td>
<td>4</td>
<td>0x0</td>
<td>[3] RX overrun interrupt, write 1 to clear</td>
</tr>
<tr>
<td>INTCLEAR</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>[2] TX overrun interrupt, write 1 to clear</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>[1] RX interrupt, write 1 to clear</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>[0] TX interrupt, write 1 to clear</td>
</tr>
<tr>
<td>BAUDDIV</td>
<td>0x010</td>
<td>RW</td>
<td>20</td>
<td>0x000000</td>
<td>[19:0] Baud rate divider, the minimum number is 16</td>
</tr>
</tbody>
</table>
### 4.3 Initialization

The definition of UART initialization is shown in Table 4-2.

**Table 4-2 UART Initialization Definition**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>UART_BaudRate</td>
<td>uint32_t</td>
<td>Max 921.6Kbit/s</td>
<td>Baud rate</td>
</tr>
<tr>
<td>UART_Mode</td>
<td>UARTMode_TypeDef</td>
<td>ENABLE/DISABLE</td>
<td>Enable/Disable TX/RX mode</td>
</tr>
<tr>
<td>UART_Int</td>
<td>UARTInt_TypeDef</td>
<td>ENABLE/DISABLE</td>
<td>Enable/Disable TX/RX interrupt</td>
</tr>
<tr>
<td>UART_Ovr</td>
<td>UARTOvr_TypeDef</td>
<td>ENABLE/DISABLE</td>
<td>Enable/Disable TX/RX overrun interrupt</td>
</tr>
<tr>
<td>UART_Hstm</td>
<td>FunctionalState</td>
<td>ENABLE/DISABLE</td>
<td>Enable/Disable TX high speed test mode</td>
</tr>
</tbody>
</table>

### 4.4 Drivers Usage

The usage of UART driver is shown in Table 4-3.

**Table 4-3 Usage of UART Drivers**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>UART_Init</td>
<td>Initialize UARTx</td>
</tr>
<tr>
<td>UART_GetRxBufferFull</td>
<td>Return UARTx RX buffer full status</td>
</tr>
<tr>
<td>UART_GetTxBufferFull</td>
<td>Return UARTx TX buffer full status</td>
</tr>
<tr>
<td>UART_GetRxBufferOverrunStatus</td>
<td>Return UARTx RX buffer overrun status</td>
</tr>
<tr>
<td>UART_GetTxBufferOverrunStatus</td>
<td>Return UARTx TX buffer overrun status</td>
</tr>
<tr>
<td>UART_ClearRxBufferOverrunStatus</td>
<td>Clear Rx buffer overrun status</td>
</tr>
<tr>
<td>UART_ClearTxBufferOverrunStatus</td>
<td>Clear Tx buffer overrun status</td>
</tr>
<tr>
<td>UART_SendChar</td>
<td>Send a character to UARTx TX buffer</td>
</tr>
<tr>
<td>UART_SendString</td>
<td>Send a string to UARTx TX buffer</td>
</tr>
<tr>
<td>UART_ReceiveChar</td>
<td>Receive a character from UARTx RX buffer</td>
</tr>
<tr>
<td>UART_GetBaudDivider</td>
<td>Return UARTx baud rate divider value</td>
</tr>
<tr>
<td>UART_GetTxIRQStatus</td>
<td>Return UARTx TX interrupt status</td>
</tr>
<tr>
<td>UART_GetRxIRQStatus</td>
<td>Return UARTx RX interrupt status</td>
</tr>
<tr>
<td>UART_ClearTxIRQ</td>
<td>Clear UARTx TX interrupt status</td>
</tr>
<tr>
<td>UART_ClearRxIRQ</td>
<td>Clear UARTx RX interrupt status</td>
</tr>
<tr>
<td>UART_GetTxOverrunIRQStatus</td>
<td>Return UARTx TX overrun interrupt status</td>
</tr>
<tr>
<td>UART_GetRxOverrunIRQStatus</td>
<td>Return UARTx RX overrun interrupt status</td>
</tr>
<tr>
<td>UART_ClearTxOverrunIRQ</td>
<td>Clear UARTx TX overrun interrupt request</td>
</tr>
<tr>
<td>UART_ClearRxOverrunIRQ</td>
<td>Clear UARTx RX overrun interrupt request</td>
</tr>
<tr>
<td>UART_SetHSTM</td>
<td>Set UARTx TX high speed test mode</td>
</tr>
<tr>
<td>UART_ClrHSTM</td>
<td>Clear UARTx TX high speed test mode</td>
</tr>
</tbody>
</table>
**4.5 Reference Design**

Gowin_EMPU_M1 supports UART reference design in ARM Keil MDK (V5.24 and above) and GOWIN MCU Designer (V1.1 and above) software environment. Get following reference designs by this [link]:

- Gowin_EMPU_M1\ref_design\MCU_RefDesign\Keil_RefDesign\uart
- Gowin_EMPU_M1\ref_design\MCU_RefDesign\Keil_RefDesign\uart0_irq
- Gowin_EMPU_M1\ref_design\MCU_RefDesign\GMD_RefDesign\cm1_uart
- Gowin_EMPU_M1\ref_design\MCU_RefDesign\GMD_RefDesign\cm1_uart0_irq
5 Timer

5.1 Features

Gowin_EMPU_M1 includes two synchronization standard timers accessed by APB bus:

- 32-bit counter
- Supports the interrupt request signal
- Supports the external input signal EXTIN enabling clock

The Timer diagram is as shown in Figure 5-1.

![Figure 5-1 Timer Diagram](image)

5.2 Registers

The definition of Timer registers is as shown in Table 5-1.

<table>
<thead>
<tr>
<th>Register Name</th>
<th>Address Offset</th>
<th>Type</th>
<th>Width</th>
<th>Initial Value</th>
<th>Description</th>
</tr>
</thead>
</table>
| CTRL          | 0x000          | RW   | 4     | 0x0           | [3] Timer interrupt enable  
|               |                |      |       |               | [2] Select external input as clock  
|               |                |      |       | [1] Select external input as enable  
|               |                |      |       | [0] Enable    |
| VALUE         | 0x004          | RW   | 32    | 0x000000000   | [31:0] Current value |
| RELOAD        | 0x008          | RW   | 32    | 0x000000000   | [31:0] Reload value, writing to |
5.3 Initialization

The definition of Timer initialization is as shown in Table 5-2.

Table 5-2 Timer Initialization Definition

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reload</td>
<td>uint32_t</td>
<td>-</td>
<td>Reload value</td>
</tr>
<tr>
<td>TIMER_Int</td>
<td>TIMERInt_TypeDef</td>
<td>SET/RESET</td>
<td>Enable/Disable interrupt</td>
</tr>
<tr>
<td>TIMER_Exti</td>
<td>TIMERExti_TypeDef</td>
<td>-</td>
<td>External input as enable or clock</td>
</tr>
</tbody>
</table>

5.4 Drivers Usage

The usage of Timer drivers is as shown in Table 5-3.

Table 5-3 Usage of Timer Drivers

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TIMER_Init</td>
<td>Initialize Timerx</td>
</tr>
<tr>
<td>TIMER_StartTimer</td>
<td>Start Timerx</td>
</tr>
<tr>
<td>TIMER_StopTimer</td>
<td>Stop Timerx</td>
</tr>
<tr>
<td>TIMER_GetIRQStatus</td>
<td>Return Timerx interrupt status</td>
</tr>
<tr>
<td>TIMER_ClearIRQ</td>
<td>Clear Timerx interrupt status</td>
</tr>
<tr>
<td>TIMER_GetReload</td>
<td>Return Timerx reload value</td>
</tr>
<tr>
<td>TIMER_SetReload</td>
<td>Set Timerx reload value</td>
</tr>
<tr>
<td>TIMER_GetValue</td>
<td>Return Timerx current value</td>
</tr>
<tr>
<td>TIMER_SetValue</td>
<td>Set Timerx current value</td>
</tr>
<tr>
<td>TIMER_EnableIRQ</td>
<td>Enable Timerx interrupt request</td>
</tr>
<tr>
<td>TIMER_DisableIRQ</td>
<td>Disable Timerx interrupt request</td>
</tr>
</tbody>
</table>

5.5 Reference Design

Gowin_EMPU_M1 supports Timer reference design in ARM Keil MDK (V5.24 and above) and GOWIN MCU Designer (V1.1 and above) software environment. Get following reference designs by this link:

- Gowin_EMPU_M1\ref_design\MCU_RefDesign\Keil_RefDesign\timer
- Gowin_EMPU_M1\ref_design\MCU_RefDesign\GMD_RefDesign\cm1_timer
Gowin_EMPU_M1 includes one Watch Dog accessed by APB bus:

- A 32-bit down-counter initialized by the LOAD register;
- Support interrupt request;
- When the clock is enabled, the counter is decremented by the posedge of the WDOGCLK signal;
- Monitor interrupts and a reset request is generated and the counter is stopped when the counter is decremented to 0;
- Respond to the software reset caused by software crashes, provide the software reset method;

The operation of Watch Dog is as shown in Figure 6-1.

**Figure 6-1 Watch Dog Operation**
6.2 Registers

The definition of Watch Dog registers is as shown in Table 6-1.

Table 6-1 Definition of Watch Dog Registers

<table>
<thead>
<tr>
<th>Register Name</th>
<th>Address Offset</th>
<th>Type</th>
<th>Width</th>
<th>Initial Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>LOAD</td>
<td>0x00</td>
<td>RW</td>
<td>32</td>
<td>0xFFFFFFFF</td>
<td>The value from which the counter is to decrement</td>
</tr>
<tr>
<td>VALUE</td>
<td>0x04</td>
<td>RO</td>
<td>32</td>
<td>0xFFFFFFFF</td>
<td>The current value of the decrementing counter</td>
</tr>
<tr>
<td>CTRL</td>
<td>0x08</td>
<td>RW</td>
<td>2</td>
<td>0x0</td>
<td>[1] Enable reset output [0] Enable the interrupt</td>
</tr>
<tr>
<td>INTCLR</td>
<td>0x0C</td>
<td>WO</td>
<td>–</td>
<td>–</td>
<td>Clear the watchdog interrupt and reloads the counter</td>
</tr>
<tr>
<td>RIS</td>
<td>0x10</td>
<td>RO</td>
<td>1</td>
<td>0x0</td>
<td>Raw interrupt status from the counter</td>
</tr>
<tr>
<td>MIS</td>
<td>0x14</td>
<td>RO</td>
<td>1</td>
<td>0x0</td>
<td>Enable interrupt status from the counter</td>
</tr>
<tr>
<td>RESERVED</td>
<td>0xC00-0x014</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>Reserved</td>
</tr>
<tr>
<td>LOCK</td>
<td>0xC00</td>
<td>RW</td>
<td>32</td>
<td>0x00000000</td>
<td>[32:1] Enable register writes [0] Register write enable status</td>
</tr>
<tr>
<td>RESERVED</td>
<td>0xF00-0xC00</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>Reserved</td>
</tr>
<tr>
<td>ITCR</td>
<td>0xF00</td>
<td>RW</td>
<td>1</td>
<td>0x0</td>
<td>Integration test mode enable</td>
</tr>
<tr>
<td>ITOP</td>
<td>0xF04</td>
<td>WO</td>
<td>2</td>
<td>0x0</td>
<td>[1] Integration test WDOGRES value [0] Integration test WDOGINT value</td>
</tr>
</tbody>
</table>

6.3 Initialization

The initialization of Watch Dogs is as shown in Table 6-2.

Table 6-2 Initialization of Watch Dogs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>WDOG_Reload</td>
<td>uint32_t</td>
<td>–</td>
<td>Reload value</td>
</tr>
<tr>
<td>WDOG_Lock</td>
<td>WDOGLock_TypeDef</td>
<td>SET/RESET</td>
<td>Enable/Disable lock register write access</td>
</tr>
<tr>
<td>WDOG_Res</td>
<td>WDOGRes_TypeDef</td>
<td>SET/RESET</td>
<td>Enable/Disable reset flag</td>
</tr>
<tr>
<td>WDOG_Int</td>
<td>WDOGInt_TypeDef</td>
<td>SET/RESET</td>
<td>Enable/Disable interrupt flag</td>
</tr>
<tr>
<td>WDOG_ITMode</td>
<td>WDOGMode_TypeDef</td>
<td>SET/RESET</td>
<td>Enable/Disable integration test mode flag</td>
</tr>
</tbody>
</table>
6.4 Drivers Usage

The usage of Watch Dog drivers is as shown in Table 6-3.

Table 6-3 Usage of Watch Dog Drivers

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>WDOG_Init</td>
<td>Initializes WatchDog</td>
</tr>
<tr>
<td>WDOG_RestartCounter</td>
<td>Restart watchdog counter</td>
</tr>
<tr>
<td>WDOG_GetCounterValue</td>
<td>Returns counter value</td>
</tr>
<tr>
<td>WDOG_SetResetEnable</td>
<td>Sets reset enable</td>
</tr>
<tr>
<td>WDOG_GetResStatus</td>
<td>Returns reset status</td>
</tr>
<tr>
<td>WDOG_SetIntEnable</td>
<td>Sets interrupt enable</td>
</tr>
<tr>
<td>WDOG_GetIntStatus</td>
<td>Returns interrupt enable</td>
</tr>
<tr>
<td>WDOG_ClrIntEnable</td>
<td>Clears interrupt enable</td>
</tr>
<tr>
<td>WDOG_GetRawIntStatus</td>
<td>Returns raw interrupt status</td>
</tr>
<tr>
<td>WDOG_GetMaskIntStatus</td>
<td>Returns masked interrupt status</td>
</tr>
<tr>
<td>WDOG_LockWriteAccess</td>
<td>Disable write access all registers</td>
</tr>
<tr>
<td>WDOG_UnlockWriteAccess</td>
<td>Enable write access all registers</td>
</tr>
<tr>
<td>WDOG_SetITModeEnable</td>
<td>Sets integration test mode enable</td>
</tr>
<tr>
<td>WDOG_ClrITModeEnable</td>
<td>Clears integration test mode enable</td>
</tr>
<tr>
<td>WDOG_GetITModeStatus</td>
<td>Returns integration test mode status</td>
</tr>
<tr>
<td>WDOG_SetTOP</td>
<td>Sets integration test output reset or interrupt</td>
</tr>
<tr>
<td>WDOG_GetTOPResStatus</td>
<td>Returns integration test output reset status</td>
</tr>
<tr>
<td>WDOG_GetTOPIntStatus</td>
<td>Returns integration test output interrupt status</td>
</tr>
<tr>
<td>WDOG_ClrTOP</td>
<td>Clears integration test output reset or interrupt</td>
</tr>
</tbody>
</table>

6.5 Reference Design

Gowin_EMPU_M1 supports Watch Dog reference design in ARM Keil MDK (V5.24 and above) and GOWIN MCU Designer (V1.1 and above) software environment. Get following reference designs by this link:

- Gowin_EMPU_M1\ref_design\MCU_RefDesign\Keil_RefDesign\watchdog
- Gowin_EMPU_M1\ref_design\MCU_RefDesign\GMD_RefDesign\cm1_watchdog
7 GPIO

7.1 Features

Gowin_EMPU_M1 includes a GPIO module with a 16-bit input and output interface accessed by AHB bus:

- Connected with FPGA Fabric
- Supports bit mask
- Supports pin multiplexing

The GPIO diagram is as shown in Figure 7-1.

Figure 7-1 GPIO Block

7.2 Registers

The definition of GPIO registers is as shown in Table 7-1.

<table>
<thead>
<tr>
<th>Register Name</th>
<th>Address Offset</th>
<th>Type</th>
<th>Width</th>
<th>Initial Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DATA</td>
<td>0x0000</td>
<td>RW</td>
<td>16</td>
<td>0x----</td>
<td>[15:0] Data value Read Sampled at pin Write to data output register Read back value goes through double flip-flop</td>
</tr>
<tr>
<td>Register Name</td>
<td>Address Offset</td>
<td>Type</td>
<td>Width</td>
<td>Initial Value</td>
<td>Description</td>
</tr>
<tr>
<td>---------------</td>
<td>----------------</td>
<td>------</td>
<td>-------</td>
<td>---------------</td>
<td>-------------</td>
</tr>
</tbody>
</table>
| DATAOUT       | 0x0004         | RW   | 16    | 0x0000        | [15:0] Data output register value
Read current value of data output register
Write to data output register
synchronize logic with delay of two cycles |
| RESERVED      | 0x0008 -0x000C | -    | -     | -             | Reserved |
| OUTENSET      | 0x0010         | RW   | 16    | 0x0000        | [15:0] Output enable set
Write 1 to set the output enable bit
Write 0 no effect
Read back 0 indicates the signal direction as input
1 indicates the signal direction as output |
| OUTENCLR      | 0x0014         | RW   | 16    | 0x0000        | [15:0] Output enable clear
Write 1 to clear the output enable bit
Write 0 no effect
Read back 0 indicates the signal direction as input
1 indicates the signal direction as output |
| ALTFUNCSET    | 0x0018         | RW   | 16    | 0x0000        | [15:0] Alternative function set
Write 1 to set the ALTFUNC bit
Write 0 no effect
Read back 0 for I/O
1 for an alternate function |
| ALTFUNCCLR    | 0x001C         | RW   | 16    | 0x0000        | [15:0] Alternative function clear
Write 1 to clear the ALTFUNC bit
Write 0 no effect
Read back 0 for I/O
1 for an alternate function |
| INTENSET      | 0x0020         | RW   | 16    | 0x0000        | [15:0] Interrupt enable set
Write 1 to set the enable bit
Write 0 no effect
Read back 0 indicates interrupt disabled
1 indicates interrupt enabled |
<p>| INTENCLR      | 0x0024         | RW   | 16    | 0x0000        | [15:0] Interrupt enable |</p>
<table>
<thead>
<tr>
<th>Register Name</th>
<th>Address Offset</th>
<th>Type</th>
<th>Width</th>
<th>Initial Value</th>
<th>Description</th>
</tr>
</thead>
</table>
| clear         |                |      |       |               | Write 1 to clear the enable bit  
|               |                |      |       |               | Write 0 no effect  
|               |                |      |       |               | Read back 0 indicates interrupt disabled  
|               |                |      |       |               | 1 indicates interrupt enabled |
| INTTYPESET    | 0x0028         | RW   | 16    | 0x0000        | [15:0] Interrupt type set  
|               |                |      |       |               | Write 1 to set the interrupt type bit  
|               |                |      |       |               | Write 0 no effect  
|               |                |      |       |               | Read back 0 for LOW/HIGH level  
|               |                |      |       |               | 1 for falling edge or rising edge |
| INTTYPECLR    | 0x002C         | RW   | 16    | 0x0000        | [15:0] Interrupt type clear  
|               |                |      |       |               | Write 1 to clear the interrupt type bit  
|               |                |      |       |               | Write 0 no effect  
|               |                |      |       |               | Read back 0 for LOW/HIGH level  
|               |                |      |       |               | 1 for falling edge or rising edge |
| INTPOLSET     | 0x0030         | RW   | 16    | 0x0000        | [15:0] Polarity-level,edge IRQ config  
|               |                |      |       |               | Write 1 to set the interrupt polarity bit  
|               |                |      |       |               | Write 0 no effect  
|               |                |      |       |               | Read back 0 for LOW level or falling edge  
|               |                |      |       |               | 1 for HIGH level or rising edge |
| INTPOLCLR     | 0x0034         | RW   | 16    | 0x0000        | [15:0] Polarity-level,edge IRQ config  
|               |                |      |       |               | Write 1 to clear the interrupt polarity bit  
|               |                |      |       |               | Write 0 no effect  
|               |                |      |       |               | Read back 0 for LOW level or falling edge  
|               |                |      |       |               | 1 for HIGH level or rising edge |
| INTSTATUS     | 0x0038         | RW   | 16    | 0x0000        | [15:0] Write IRQ status clear register  
| /INTCLEAR     |                |      |       |               | Write 1 to clear interrupt request  
|               |                |      |       |               | Write 0 no effect  
<p>|               |                |      |       |               | Read back IRQ status register |</p>
<table>
<thead>
<tr>
<th>Register Name</th>
<th>Address Offset</th>
<th>Type</th>
<th>Width</th>
<th>Initial Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MASKLOWBYTE</td>
<td>0x0400 -0x07FC</td>
<td>RW</td>
<td>16</td>
<td>0x----</td>
<td>Lower 8-bits masked access [9:2] of the address value are used as enable bit mask for the access [15:8] not used [7:0] Data for lower byte access, with [9:2] of address value used as enable mask for each bit [7:0] not used</td>
</tr>
<tr>
<td>MASKHIGHBYTE</td>
<td>0x0800 -0x0BFC</td>
<td>RW</td>
<td>16</td>
<td>0x----</td>
<td>Higher 8-bits masked access [9:2] of the address value are used as enable bit mask for the access [15:8] Data for higher byte access, with [9:2] of address value used as enable mask for each bit [7:0] not used</td>
</tr>
<tr>
<td>RESERVED</td>
<td>0x0C00 -0x0FCF</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>Reserved</td>
</tr>
</tbody>
</table>
### 7.3 Initialization

The definition of GPIO initialization is as shown in Table 7-2.

#### Table 7-2 GPIO Initialization

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GPIO_Pin</td>
<td>uint32_t</td>
<td>GPIO_Pin_0, GPIO_Pin_1, GPIO_Pin_2, ...</td>
<td>16 bits GPIO Pins</td>
</tr>
<tr>
<td>GPIO_Mode</td>
<td>GPIOMode_TypeDef</td>
<td>GPIO_Mode_IN, GPIO_Mode_OUT, ...</td>
<td>16 bits GPIO Pins mode</td>
</tr>
<tr>
<td>GPIO_Int</td>
<td>GPIOInt_TypeDef</td>
<td>GPIO_Int_Disable, GPIO_Int_Low_Level, ...</td>
<td>16 bits GPIO Pins interrupt</td>
</tr>
</tbody>
</table>
7.4 Drivers Usage

The usage of GPIO drivers is as shown in Table 7-3.

Table 7-3 Usage of GPIO Drivers

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GPIO_Init</td>
<td>Initialize GPIOx</td>
</tr>
<tr>
<td>GPIO_SetOutEnable</td>
<td>Set GPIOx output enable</td>
</tr>
<tr>
<td>GPIOClrOutEnable</td>
<td>Clear GPIOx output enable</td>
</tr>
<tr>
<td>GPIO_GetOutEnable</td>
<td>Return GPIOx output enable</td>
</tr>
<tr>
<td>GPIO_SetBit</td>
<td>GPIO output one</td>
</tr>
<tr>
<td>GPIO_ResetBit</td>
<td>GPIO output zero</td>
</tr>
<tr>
<td>GPIO_WriteBits</td>
<td>GPIO output</td>
</tr>
<tr>
<td>GPIO_ReadBits</td>
<td>GPIO input</td>
</tr>
<tr>
<td>GPIO_SetAltFunc</td>
<td>Set GPIOx alternate function enable</td>
</tr>
<tr>
<td>GPIOClrAltFunc</td>
<td>Clear GPIOx alternate function enable</td>
</tr>
<tr>
<td>GPIO_GetAltFunc</td>
<td>Return GPIOx alternate function enable</td>
</tr>
<tr>
<td>GPIO_IntClear</td>
<td>Clear GPIOx interrupt request</td>
</tr>
<tr>
<td>GPIO_GetIntStatus</td>
<td>Return GPIOx interrupt status</td>
</tr>
<tr>
<td>GPIO_SetIntEnable</td>
<td>Set GPIOx interrupt enable</td>
</tr>
<tr>
<td></td>
<td>Return GPIOx interrupt status</td>
</tr>
<tr>
<td>GPIOClrIntEnable</td>
<td>Clear GPIOx interrupt enable</td>
</tr>
<tr>
<td></td>
<td>Return GPIOx interrupt enable</td>
</tr>
<tr>
<td>GPIO_SetIntHighLevel</td>
<td>Setup GPIOx interrupt as high level</td>
</tr>
<tr>
<td>GPIO_SetIntRisingEdge</td>
<td>Setup GPIOx interrupt as rising edge</td>
</tr>
<tr>
<td>GPIO_SetIntLowLevel</td>
<td>Setup GPIOx interrupt as low level</td>
</tr>
<tr>
<td>GPIO_SetIntFallingEdge</td>
<td>Setup GPIOx interrupt as falling edge</td>
</tr>
<tr>
<td>GPIO_MaskedWrite</td>
<td>Setup GPIOx output value using masked access</td>
</tr>
</tbody>
</table>

7.5 Reference Design

Gowin_EMPU_M1 supports GPIO reference designs in ARM Keil MDK (V5.24 and above) and GOWIN MCU Designer (V1.1 and above) software environment. Get following reference designs by this [link]:

- Gowin_EMPU_M1\ref_design\MCU_RefDesign\Keil_RefDesign\led
- Gowin_EMPU_M1\ref_design\MCU_RefDesign\Keil_RefDesign\keyscan
- Gowin_EMPU_M1\ref_design\MCU_RefDesign\GMD_RefDesign\cm1\_led
- Gowin_EMPU_M1\ref_design\MCU_RefDesign\GMD_RefDesign\cm1\_keyscan
8 I²C

8.1 Features

Gowin_EMPU_M1 includes an I²C Master module accessed by AHB bus:

- APB bus interface
- Compliant with industry standard I²C protocol
- Bus arbitration and arbitration lost detection
- Bus busy detection
- Interrupt flag generation
- Generates Start, Stop, Repeated Start and Acknowledge
- Detects Start, Stop and Repeated Start
- Supports 7-bit addressing mode

8.2 Registers

The definition of I²C Master registers is as described in Table 8-1.

<table>
<thead>
<tr>
<th>Register Name</th>
<th>Address Offset</th>
<th>Type</th>
<th>Width</th>
<th>Initial Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PRER</td>
<td>0x00</td>
<td>RW</td>
<td>32</td>
<td>0x0000FFFF</td>
<td>Clock prescale register</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>[31:15] Reserved</td>
<td>[15:0] Prescale value = sys_clk/(5*SCL)-1</td>
</tr>
<tr>
<td>CTR</td>
<td>0x04</td>
<td>RW</td>
<td>32</td>
<td>0x00000000</td>
<td>[31:8] Reserved</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>[5:0] Reserved</td>
<td></td>
</tr>
<tr>
<td>TXR</td>
<td>0x08</td>
<td>WO</td>
<td>32</td>
<td>0x00000000</td>
<td>[31:8] Reserved</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>[7:1] Next transmission data</td>
<td>[0] Data direction</td>
</tr>
<tr>
<td>RXR</td>
<td>0x0C</td>
<td>RO</td>
<td>32</td>
<td>0x00000000</td>
<td>[31:8] Reserved</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>[7:0] Last received data</td>
<td></td>
</tr>
<tr>
<td>CR</td>
<td>0x010</td>
<td>WO</td>
<td>32</td>
<td>0x00000000</td>
<td>[31:8] Reserved</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>[7] STA, Start transmission</td>
<td></td>
</tr>
</tbody>
</table>
### 8.3 Drivers Usage

The usage of I²C Master drivers is as described in Table 8-2.

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>I2C_Init</td>
<td>I²C Initialization</td>
</tr>
<tr>
<td>I2C_SendByte</td>
<td>Send a byte to I²C bus</td>
</tr>
<tr>
<td>I2C_SendBytes</td>
<td>Send multiple bytes to I²C bus</td>
</tr>
<tr>
<td>I2C_SendData</td>
<td>Send multiple bytes to I²C bus once time</td>
</tr>
<tr>
<td>I2C_ReceiveByte</td>
<td>Read a byte from I²C bus</td>
</tr>
<tr>
<td>I2C_ReadBytes</td>
<td>Read multiple bytes from I²C bus</td>
</tr>
<tr>
<td>I2C_ReceiveData</td>
<td>Read multiple bytes from I²C bus once time</td>
</tr>
<tr>
<td>I2C_Rate_Set</td>
<td>Set I²C traffic rate</td>
</tr>
<tr>
<td>I2C_Enable</td>
<td>Enable I²C bus</td>
</tr>
<tr>
<td>I2C_UnEnable</td>
<td>Disable I²C bus</td>
</tr>
<tr>
<td>I2C_InterruptOpen</td>
<td>Open I²C interrupt</td>
</tr>
<tr>
<td>I2C_InterruptClose</td>
<td>Close I²C interrupt</td>
</tr>
</tbody>
</table>

### 8.4 Reference Design

Gowin_EMPU_M1 supports I²C Master reference design in ARM Keil MDK (V5.24 and above) and GOWIN MCU Designer (V1.1 and above) software environment. Get following reference designs by this link:

- Gowin_EMPU_M1/ref_design\MCU_RefDesign\Keil_RefDesign\i2c
- Gowin_EMPU_M1/ref_design\MCU_RefDesign\GMD_RefDesign\cm1_i2c
9 SPI

9.1 Features

Gowin_EMPU_M1 includes a SPI Master accessed by AHB bus:
- APB bus interface
- Full duplex synchronous serial data transmission
- Supports Master working mode
- Supports configurable clock polarity and phase
- Configurable serial clock frequency generated by SPI
- 8 bits width for data receive register and data transmission register

9.2 Registers

The definition of SPI Master registers is as described in Table 9-1.

Table 9-1 Definition of SPI Master Registers

<table>
<thead>
<tr>
<th>Register Name</th>
<th>Address Offset</th>
<th>Type</th>
<th>Width</th>
<th>Initial Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RDATA</td>
<td>0x00</td>
<td>RO</td>
<td>32</td>
<td>0x00000000</td>
<td>Read data register [31:8] Reserved [7:0] Read data</td>
</tr>
<tr>
<td>WDATA</td>
<td>0x04</td>
<td>WO</td>
<td>32</td>
<td>0x00000000</td>
<td>Write data register [31:8] Reserved [7:0] Write data</td>
</tr>
<tr>
<td>SSMASK</td>
<td>0x0C</td>
<td>RW</td>
<td>32</td>
<td>0x00000000</td>
<td>[31:1] Reserved [0] Select and enable slave</td>
</tr>
</tbody>
</table>
### 9.3 Initialization

The definition of SPI Master initialization is as described in Table 9-2.

**Table 9-2 SPI Master Initialization**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DIRECTION</td>
<td>FunctionalState</td>
<td>ENABLE/DISABLE</td>
<td>MSB/LSB first transmission</td>
</tr>
<tr>
<td>PHASE</td>
<td>FunctionalState</td>
<td>ENABLE/DISABLE</td>
<td>Posedge/Negedge transmit data</td>
</tr>
<tr>
<td>POLARITY</td>
<td>FunctionalState</td>
<td>ENABLE/DISABLE</td>
<td>Initialize polarity to one/zero</td>
</tr>
<tr>
<td>CLKSEL</td>
<td>uint32_t</td>
<td>CLKSEL_CLK_DIV_2, CLKSEL_CLK_DIV_4, CLKSEL_CLK_DIV_6, CLKSEL_CLK_DIV_8</td>
<td>Select clock divided 2/4/6/8</td>
</tr>
</tbody>
</table>
9.4 Drivers Usage

The usage of SPI Master drivers is as described in Table 9-3.

Table 9-3 Usage of SPI Master Drivers

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SPI_Init</td>
<td>Initialize SPI</td>
</tr>
<tr>
<td>SPI_SetDirection</td>
<td>Set direction</td>
</tr>
<tr>
<td>SPI_ClrDirection</td>
<td>Clear direction</td>
</tr>
<tr>
<td>SPI_GetDirection</td>
<td>Return direction</td>
</tr>
<tr>
<td>SPI_SetPhase</td>
<td>Set phase</td>
</tr>
<tr>
<td>SPI_ClrPhase</td>
<td>Clear phase</td>
</tr>
<tr>
<td>SPI_GetPhase</td>
<td>Return phase</td>
</tr>
<tr>
<td>SPI_SetPolarity</td>
<td>Set polarity</td>
</tr>
<tr>
<td>SPI_ClrPolarity</td>
<td>Clear polarity</td>
</tr>
<tr>
<td>SPI_GetPolarity</td>
<td>Return polarity</td>
</tr>
<tr>
<td>SPI_SetClkSel</td>
<td>Set clock selection</td>
</tr>
<tr>
<td>SPI_GetClkSel</td>
<td>Return clock selection</td>
</tr>
<tr>
<td>SPI_GetToeStatus</td>
<td>Read transmit overrun error status</td>
</tr>
<tr>
<td>SPI_GetRoeStatus</td>
<td>Read receive overrun error status</td>
</tr>
<tr>
<td>SPI_GetTmtStatus</td>
<td>Read transmitting status</td>
</tr>
<tr>
<td>SPI_GetTrdyStatus</td>
<td>Read transmit ready status</td>
</tr>
<tr>
<td>SPI_GetRrdyStatus</td>
<td>Read receive ready error status</td>
</tr>
<tr>
<td>SPI_GetErrStatus</td>
<td>Read error status</td>
</tr>
<tr>
<td>SPI_ClrToeStatus</td>
<td>Clear transmit overrun error status</td>
</tr>
<tr>
<td>SPI_ClrRoeStatus</td>
<td>Clear receive overrun error status</td>
</tr>
<tr>
<td>SPI_ClrErrStatus</td>
<td>Clear error status</td>
</tr>
<tr>
<td>SPI_WriteData</td>
<td>Write data</td>
</tr>
<tr>
<td>SPI_ReadData</td>
<td>Read data</td>
</tr>
<tr>
<td>SPI_Select_Slave</td>
<td>Select slave</td>
</tr>
</tbody>
</table>

9.5 Reference Design

Gowin_EMPU_M1 supports SPI Master reference design in ARM Keil MDK (V5.24 and above) and GOWIN MCU Designer (V1.1 and above) software environment. Get following reference designs by this link:

- Gowin_EMPU_M1\ref_design\MCU_RefDesign\Keil_RefDesign\spi
- Gowin_EMPU_M1\ref_design\MCU_RefDesign\GMD_RefDesign\cm1_spi
10 Real-time Clock

10.1 Features

Gowin_EMPU_M1 includes a 32-bit real-time clock (RTC) module accessed by AHB bus:

- APB bus interface
- 32-bit counter
- 32-bit Match register
- 32-bit comparator

MCU reads data, control, and status messages via APB bus interface and RTC. At the posedge of continuous input clock CLK1HZ, 32 bits counter increases.

This counter is not synchronous and can not be overloaded. When system resets, this counter counts from 1 to the max. value (0xFFFFFFFF) and then goes back to 0 and keeps increasing.

Realize RTC load or update via the write load register RTC_LOAD_VALUE

Obtain RTC current clock via the read data register RTC_CURRENT_DATA

Program Match register via the register of write RTC_MATCH_VALUE

RTC diagram is as shown in Figure 10-1.
10 Real-time Clock

10.2 Registers

The definition of RTC registers is as described in Table 10-1.

<table>
<thead>
<tr>
<th>Register Name</th>
<th>Address Offset</th>
<th>Type</th>
<th>Width</th>
<th>Initial Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RTC_CURRENT_DATA</td>
<td>0x000</td>
<td>RO</td>
<td>32</td>
<td>0x00000000</td>
<td>Data Register [31:0] Current value</td>
</tr>
<tr>
<td>RTC_MATCH_VALUE</td>
<td>0x004</td>
<td>RW</td>
<td>32</td>
<td>0x00000000</td>
<td>Match Register If current value equals match register’s value, generate interrupt [31:0] Match data</td>
</tr>
<tr>
<td>RTC_LOAD_VALUE</td>
<td>0x008</td>
<td>RW</td>
<td>32</td>
<td>0x00000000</td>
<td>Load Register Initialized value, start counter based on this value [31:0] Load data</td>
</tr>
<tr>
<td>RTC_CONTROLLER_REG</td>
<td>0x00C</td>
<td>RW</td>
<td>32</td>
<td>0x00000000</td>
<td>Control Register Start RTC counter [31:1] Reserved [0] Start RTC counter</td>
</tr>
<tr>
<td>RTC_IMSC</td>
<td>0x010</td>
<td>RW</td>
<td>32</td>
<td>0x00000000</td>
<td>Interrupt mask set and clear register Enable or disable interrupt [31:1] Reserved [0] Enable interrupt</td>
</tr>
<tr>
<td>RTC_RIS</td>
<td>0x014</td>
<td>RO</td>
<td>32</td>
<td>0x00000000</td>
<td>Raw interrupt status register Get current raw unmasked interrupt status [31:1] Reserved [0] Current raw unmasked</td>
</tr>
</tbody>
</table>
### 10.3 Drivers Usage

The usage of RTC drivers is as described in Table 10-2.

**Table 10-2 Usage of RTC Drivers**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RTC_init</td>
<td>Initialize RTC</td>
</tr>
<tr>
<td>Get_Current_Value</td>
<td>Get RTC current value of data register</td>
</tr>
<tr>
<td>Set_Match_Value</td>
<td>Set RTC match value of match register</td>
</tr>
<tr>
<td>Get_Match_Value</td>
<td>Get RTC match value of match register</td>
</tr>
<tr>
<td>Set_Load_Value</td>
<td>Set RTC load value of load register</td>
</tr>
<tr>
<td>Get_Load_Value</td>
<td>Get RTC load value of load register</td>
</tr>
<tr>
<td>Start_RTC</td>
<td>Start RTC counter</td>
</tr>
<tr>
<td>Close_RTC</td>
<td>Close RTC counter</td>
</tr>
<tr>
<td>RTC_Inter_Mask_Set</td>
<td>Set RTC interrupt mask</td>
</tr>
<tr>
<td>Get_RTC_Control_value</td>
<td>Get value of control register</td>
</tr>
<tr>
<td>RTC_Inter_Mask_Clr</td>
<td>Clear RTC interrupt mask</td>
</tr>
<tr>
<td>Get_RTC_Inter_Mask_value</td>
<td>Get RTC interrupt mask</td>
</tr>
<tr>
<td>Clear_RTC_interrupt</td>
<td>Clear RTC interrupt</td>
</tr>
</tbody>
</table>

### 10.4 Reference Design

Gowin_EMPU_M1 supports RTC reference design in ARM Keil MDK (V5.24 and above) and GOWIN MCU Designer (V1.1 and above) software environment. Get following reference designs by this [link]:

- Gowin_EMPU_M1\ref_design\MCU_RefDesign\Keil_RefDesign\rtc
- Gowin_EMPU_M1\ref_design\MCU_RefDesign\GMD_RefDesign\cm1_rtc
11 True Random Number Generator

11.1 Features

Gowin_EMPU_M1 includes a 32-bit true random number generator (TRNG) module accessed by AHB bus:

- Digital logic generates and adopts a true random number bitstream;
- Includes an internal entropy source based on a digital inverter chain;
- If MCU core runs at 200MHz, a 10K bits/s entropy can be generated;
- APB bus interface.

11.2 Registers

The definition of TRNG registers is as described in Table 11-1.

<table>
<thead>
<tr>
<th>Register Name</th>
<th>Address Offset</th>
<th>Type</th>
<th>Width</th>
<th>Initial Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RESERVE1</td>
<td>0x000-0x0FC</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>Reserved</td>
</tr>
<tr>
<td>Register Name</td>
<td>Address Offset</td>
<td>Type</td>
<td>Width</td>
<td>Initial Value</td>
<td>Description</td>
</tr>
<tr>
<td>----------------</td>
<td>----------------</td>
<td>------</td>
<td>-------</td>
<td>---------------</td>
<td>-------------</td>
</tr>
<tr>
<td>RNG_ICR</td>
<td>0x108</td>
<td>WO</td>
<td>32</td>
<td>0x00000000</td>
<td>Interrupt clear register</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>[0] Set to 1 When 192 bits have been collected, and EHR_DATA[0-5] registers are ready to be read</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>[31:4] Reserved</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>[3] Clear a Von Neumann error</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>[2] Clear a CRNGT error</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>[1] Software cannot clear this bit, only a TRNG reset can clear this bit</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>[0] Set to 1 after EHR_DATA[0-5] have been read</td>
</tr>
<tr>
<td>TRNG_CONFIG</td>
<td>0x10C</td>
<td>RW</td>
<td>32</td>
<td>0x00000000</td>
<td>Configuration register</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>[31:2] Reserved</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>[1:0] Selects the number of inverters:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>00 = Selects the shortest inverter chain length</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>01 = Selects the short inverter chain length</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>10 = Selects the long inverter chain length</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>11 = Selects the longest inverter chain length</td>
</tr>
<tr>
<td>TRNG_VALID</td>
<td>0x110</td>
<td>RO</td>
<td>32</td>
<td>0x00000000</td>
<td>Valid register</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>[31:1] Reserved</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>[0] TRNG is complete, data can be read from EHR_DATA[0-5]</td>
</tr>
<tr>
<td>EHR_DATA0</td>
<td>0x114—x128</td>
<td>RO</td>
<td>32</td>
<td>0x00000000</td>
<td>Entropy holding register data register</td>
</tr>
<tr>
<td>EHR_DATA1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Return 32 bits from the 192-bit EHR DATA0 returns bits[31:0]</td>
</tr>
<tr>
<td>EHR_DATA2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>DATA1 returns bits[63:32]</td>
</tr>
<tr>
<td>EHR_DATA3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>DATA2 returns bits[95:64]</td>
</tr>
<tr>
<td>EHR_DATA4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>DATA3 returns bits[127:96]</td>
</tr>
<tr>
<td>EHR_DATA5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>DATA4 returns bits[159:128]</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>DATA5 returns bits[191:160]</td>
</tr>
<tr>
<td>Register Name</td>
<td>Address Offset</td>
<td>Type</td>
<td>Width</td>
<td>Initial Value</td>
<td>Description</td>
</tr>
<tr>
<td>----------------------------</td>
<td>----------------</td>
<td>------</td>
<td>-------</td>
<td>---------------</td>
<td>--------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>RND_SOURCE_ENABLE</td>
<td>0x12C</td>
<td>RW</td>
<td>32</td>
<td>0x00000000</td>
<td>Random source enable register [31:1] Reserved [0] 1 = enable entropy source; 0 = disable entropy source</td>
</tr>
<tr>
<td>SAMPLE_CNT1</td>
<td>0x130</td>
<td>RW</td>
<td>32</td>
<td>0x0000FFFF</td>
<td>Sample count register [31:0] Sets the number of rng_clk cycles</td>
</tr>
<tr>
<td>AUTOCORR_STATISTIC</td>
<td>0x134</td>
<td>RW</td>
<td>32</td>
<td>0x00000000</td>
<td>Autocorrelation register [31:22] Reserved [21:14] Count each time an autocorrelation test fails [13:0] Count each time an autocorrelation test starts</td>
</tr>
<tr>
<td>RESERVE2</td>
<td>0x13C</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>Reserved</td>
</tr>
<tr>
<td>TRNG_SW_RESET</td>
<td>0x140</td>
<td>WO</td>
<td>32</td>
<td>0x00000000</td>
<td>Reset register [31:1] Reserved [0] Writing 1 to this register causes an internal TRNG reset</td>
</tr>
<tr>
<td>RESERVE3</td>
<td>0x144-0x1B4</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>Reserved</td>
</tr>
<tr>
<td>TRNG_BUSY</td>
<td>0x1B8</td>
<td>RO</td>
<td>32</td>
<td>0x00000000</td>
<td>Busy register [31:1] Reserved [0] Reflects the status of rng_busy signal</td>
</tr>
<tr>
<td>RST_BIT_COUNT</td>
<td>0x1BC</td>
<td>WO</td>
<td>32</td>
<td>0x00000000</td>
<td>Reset bits counter register [31:1] Reserved [0] Write any value to this bit resets the bits counter and TRNG valid registers</td>
</tr>
<tr>
<td>RESERVE4</td>
<td>0x1C0-0x1DC</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>Reserved</td>
</tr>
<tr>
<td>RNG_BIST_CNT R0</td>
<td>0x1E0-0x1E8</td>
<td>RO</td>
<td>32</td>
<td>0x00000000</td>
<td>BIST counter registers Return the collected BIST results</td>
</tr>
<tr>
<td>RNG_BIST_CNT</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
11.3 Drivers Usage

The usage of TRNG drivers is as described in Table 11-2.

Table 11-2 Usage of TRNG Driver

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Init_TRNG</td>
<td>Initialized TRNG</td>
</tr>
<tr>
<td>Set_Interrupt_Mask</td>
<td>Set interrupt mask</td>
</tr>
<tr>
<td>Get_Int_State</td>
<td>Get interrupt status</td>
</tr>
<tr>
<td>Clear_Int</td>
<td>Clear interrupt</td>
</tr>
<tr>
<td>Set_Config</td>
<td>Set config register</td>
</tr>
<tr>
<td>Get_EHR_Data</td>
<td>Get Entropy holding data</td>
</tr>
<tr>
<td>Set_Random_Source_Enable</td>
<td>Set random source enable register</td>
</tr>
<tr>
<td>Clr_Random_Source_Enable</td>
<td>Clear random source enable register</td>
</tr>
<tr>
<td>Set_Sample_Count</td>
<td>Set sample count register</td>
</tr>
<tr>
<td>Trng_SW_Reset</td>
<td>Reset TRNG</td>
</tr>
<tr>
<td>Get_TRNG_State</td>
<td>Get TRNG state</td>
</tr>
<tr>
<td>Reset_Bit_Count</td>
<td>Reset bit count register</td>
</tr>
<tr>
<td>Get_Bit.Counter</td>
<td>Get bits count register</td>
</tr>
<tr>
<td>Set_Debug_Control</td>
<td>Set debug control register</td>
</tr>
<tr>
<td>Fail_Start_State_times</td>
<td>Get autocorrelation register</td>
</tr>
<tr>
<td>Clr_Fail_Start_State_register</td>
<td>Clear autocorrelation register</td>
</tr>
</tbody>
</table>

11.4 Reference Design

Gowin_EMPU_M1 supports TRNG reference design in ARM Keil MDK (V5.24 and above) and GOWIN MCU Designer (V1.1 and above) software environment. Get following reference designs by this link:

- Gowin_EMPU_M1\ref_design\MCU_RefDesign\Keil_RefDesign\trng
- Gowin_EMPU_M1\ref_design\MCU_RefDesign\GMD_RefDesign\cm1_trng
12 Dual Timer

12.1 Features

Gowin_EMPU_M1 includes a 32-bit and 16-bit DualTimer module accessed by AHB bus:

- APB bus interface
- Includes two programmable 32-bit or 16-bit down-counter
- Generates interrupt when the down-counter reaches 0

12.2 Registers

The definition of down-counter registers is as described in Table 12-1.

<table>
<thead>
<tr>
<th>Register Name</th>
<th>Address Offset</th>
<th>Type</th>
<th>Width</th>
<th>Initial Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TIMER1LOAD</td>
<td>0x00</td>
<td>RW</td>
<td>32</td>
<td>0x00000000</td>
<td>Timer1 load register [31:0] Timer1 load value</td>
</tr>
<tr>
<td>TIMER1VALUE</td>
<td>0x04</td>
<td>RO</td>
<td>32</td>
<td>0xFFFFFFF</td>
<td>Timer1 current value register [31:0] Timer1 current value</td>
</tr>
<tr>
<td>Register Name</td>
<td>Address Offset</td>
<td>Type</td>
<td>Width</td>
<td>Initial Value</td>
<td>Description</td>
</tr>
<tr>
<td>--------------------</td>
<td>----------------</td>
<td>------</td>
<td>-------</td>
<td>---------------</td>
<td>-------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>TIMER1INTCLR</td>
<td>0x0C</td>
<td>WO</td>
<td>-</td>
<td>-</td>
<td>Timer1 interrupt clear register</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Any write clears the interrupt output of the counter</td>
</tr>
<tr>
<td>TIMER1RIS</td>
<td>0x10</td>
<td>RO</td>
<td>32</td>
<td>0x00000000</td>
<td>Timer1 raw interrupt status register</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>[31:1] Reserved</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>[0] Raw interrupt status from the counter</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TIMER1MIS</td>
<td>0x14</td>
<td>RO</td>
<td>32</td>
<td>0x00000000</td>
<td>Timer1 interrupt status register</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>[31:1] Reserved</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>[0] Enable interrupt status from the counter</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TIMER1BGLOAD</td>
<td>0x18</td>
<td>RW</td>
<td>32</td>
<td>0x00000000</td>
<td>Timer1 background load register</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>[31:0] The value used to reload the counter</td>
</tr>
<tr>
<td>RESERVE1</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>Reserved</td>
</tr>
<tr>
<td>TIMER2LOAD</td>
<td>0x00</td>
<td>RW</td>
<td>32</td>
<td>0x00000000</td>
<td>Timer2 load register</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>[31:0] Timer2 load value</td>
</tr>
<tr>
<td>TIMER2VALUE</td>
<td>0x04</td>
<td>RO</td>
<td>32</td>
<td>0xFFFFFFFF</td>
<td>Timer2 current value register</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>[31:0] Timer2 current value</td>
</tr>
<tr>
<td>TIMER2CONTROL</td>
<td>0x08</td>
<td>RW</td>
<td>32</td>
<td>0x00000020</td>
<td>Timer2 control register</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>[31:8] Reserved</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>[7] Timer enable</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>[6] Timer mode</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>[5] Interrupt enable</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>[4] Reserved</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>[3:2] Timer prescale</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>00 = clock is divided by 1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>01 = clock is divided by 16</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>10 = clock is divided by 256</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>11 Undefined</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>[1] Timer size</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0 = 16-bit counter, default</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1 = 32-bit counter</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>[0] One-shot count</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0 = wrapping mode, default</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1 = one-shot mode</td>
</tr>
<tr>
<td>TIMER2INTCLR</td>
<td>0x0C</td>
<td>WO</td>
<td>-</td>
<td>-</td>
<td>Timer2 interrupt clear register</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Any write clears the interrupt output of the counter</td>
</tr>
<tr>
<td>TIMER2RIS</td>
<td>0x10</td>
<td>RO</td>
<td>32</td>
<td>0x00000000</td>
<td>Timer2 raw interrupt status register</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>[31:1] Reserved</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>[0] Raw interrupt status from the counter</td>
</tr>
<tr>
<td>TIMER2MIS</td>
<td>0x14</td>
<td>RO</td>
<td>32</td>
<td>0x00000000</td>
<td>Timer2 interrupt status register</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>[31:1] Reserved</td>
</tr>
</tbody>
</table>
### 12.3 Drivers Usage

The usage of DualTimer drivers is as described in Table 12-2.

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DUALTIMER1_Init</td>
<td>Initialized DualTimer1</td>
</tr>
<tr>
<td>DUALTIMER2_Init</td>
<td>Initialized DualTimer2</td>
</tr>
<tr>
<td>Clear_DUALATIMER_interrupt</td>
<td>Clear DualTimer interrupt</td>
</tr>
<tr>
<td>Dtimer_MODE_function</td>
<td>Set timer mode of DualTimer1 or DualTimer2</td>
</tr>
<tr>
<td>Dtimer_PRE_function</td>
<td>Set timer prescale of DualTimer1 or DualTimer2</td>
</tr>
<tr>
<td>INIT_NUM_load_function</td>
<td>Set load value of DualTimer1 or DualTimer2</td>
</tr>
<tr>
<td>ENABLE_interrupt_Dtimer_function</td>
<td>Enable interrupt of DualTimer1 or DualTimer2</td>
</tr>
<tr>
<td>TIMER_SIZE_function</td>
<td>Set timer size of DualTimer1 or DualTimer2</td>
</tr>
<tr>
<td>ENABLE_Dtimer_function</td>
<td>Enable DualTimer1 or DualTimer2</td>
</tr>
<tr>
<td>Get_DUALATIMER_interrupt_num</td>
<td>Get timer ID of DualTimer1 or DualTimer2</td>
</tr>
</tbody>
</table>

### 12.4 Reference Design

Gowin_EMPU_M1 supports DualTimer reference design in ARM Keil MDK (V5.24 and above) and GOWIN MCU Designer (V1.1 and above) software environment. Get following reference designs by this link:

- Gowin_EMPU_M1\ref_design\MCU_RefDesign\Keil_RefDesign\dualtimer
- Gowin_EMPU_M1\ref_design\MCU_RefDesign\GMD_RefDesign\cm1_dualtimer
13 SD Card

13.1 Features

Gowin_EMPU_M1 includes one SD-Card module accessed by one APB:

- Supports SD/MMC cards
- Supports hardware initialization of cards
- Simple SPI bus access
- Supports block read and write
- Embedded receiving and transmitting buffer of 512 bytes
- APB bus interface
- Independent clocks for APB interface and SPI core logic
- The data transmitting speed close to the max. speed of SD/MMC cards

13.2 Registers

The definition of SD-Card registers is as described in Table 13-1.

<table>
<thead>
<tr>
<th>Register Name</th>
<th>Address Offset</th>
<th>Type</th>
<th>Width</th>
<th>Initial Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SPI_MASTER_VERSION</td>
<td>0x000</td>
<td>RW</td>
<td>8</td>
<td>0x00</td>
<td>SPI master version register [7:4] Major revision number [3:0] Minor revision number</td>
</tr>
<tr>
<td>SPI_MASTER_CONTROL</td>
<td>0x001</td>
<td>WO</td>
<td>8</td>
<td>0x00</td>
<td>SPI master control register [7:1] Reserved [0] Reset core logic and register 1 = Reset core logic and register, self clearing</td>
</tr>
<tr>
<td>TRANS_TYPE</td>
<td>0x002</td>
<td>RW</td>
<td>8</td>
<td>0x00</td>
<td>Transaction type register [7:2] Reserved [1:0] Set the transaction type 00 = Direct access 01 = Initialized SD 10 = Read SD block</td>
</tr>
<tr>
<td>Register Name</td>
<td>Address Offset</td>
<td>Type</td>
<td>Width</td>
<td>Initial Value</td>
<td>Description</td>
</tr>
<tr>
<td>--------------------</td>
<td>----------------</td>
<td>------</td>
<td>-------</td>
<td>---------------</td>
<td>-------------</td>
</tr>
<tr>
<td>TRANS_CTRL</td>
<td>0x003</td>
<td>WO</td>
<td>8</td>
<td>0x00</td>
<td>Transaction control register [7:1] Reserved [0] Start transaction 1 = Start transaction, self clearing</td>
</tr>
<tr>
<td>TRANS_STS</td>
<td>0x004</td>
<td>RO</td>
<td>8</td>
<td>-</td>
<td>Transaction status register [7:1] Reserved [0] Transaction busy 1 = Transaction busy</td>
</tr>
<tr>
<td>TRANS_ERROR</td>
<td>0x005</td>
<td>RO</td>
<td>8</td>
<td>-</td>
<td>Transaction error register [7:6] Reserved [5:4] SD write error 00 = Write no error 01 = Write command error 10 = Write data error 11 = Write busy error [3:2] SD read error 00 = Read no error 01 = Read command error 10 = Read token error 11 = Read busy error [1:0] SD initialize error 00 = Initialize no error 01 = Initialize command 0 error 10 = Initialize command 1 error</td>
</tr>
<tr>
<td>DIRECT_ACCESS</td>
<td>0x006</td>
<td>RW</td>
<td>8</td>
<td>0x00 / -</td>
<td>Data direct access register [7:0] Transmit data Set TX_DATA prior to starting a DIRECT_ACCESS transaction [7:0] Receive data Read RX_DATA after completing a DIRECT_ACCESS transaction</td>
</tr>
<tr>
<td>SD_ADDR_7_0</td>
<td>0x007</td>
<td>RW</td>
<td>8</td>
<td>0x00</td>
<td>SD address[7:0] bits register [7:0] SD_ADDR[7:0]</td>
</tr>
<tr>
<td>SD_ADDR_15_8</td>
<td>0x008</td>
<td>RW</td>
<td>8</td>
<td>0x00</td>
<td>SD address[15:8] bits register [7:0] SD_ADDR[15:8]</td>
</tr>
<tr>
<td>SD_ADDR_23_16</td>
<td>0x009</td>
<td>RW</td>
<td>8</td>
<td>0x00</td>
<td>SD address[23:16] bits register [7:0] SD_ADDR[23:16]</td>
</tr>
<tr>
<td>SD_ADDR_31_24</td>
<td>0x00A</td>
<td>RW</td>
<td>8</td>
<td>0x00</td>
<td>SD address[31:24] bits register [7:0] SD_ADDR[31:24]</td>
</tr>
<tr>
<td>SPI_CLK_DEL</td>
<td>0x00B</td>
<td>RW</td>
<td>8</td>
<td>0x00</td>
<td>SPI clock control register [7:0] Control the frequency of the SPI_CLK after SD initialization is completed</td>
</tr>
<tr>
<td>Register Name</td>
<td>Address Offset</td>
<td>Type</td>
<td>Width</td>
<td>Initial Value</td>
<td>Description</td>
</tr>
<tr>
<td>--------------------------</td>
<td>----------------</td>
<td>------</td>
<td>-------</td>
<td>---------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>RESERVED0</td>
<td>0x00C-0x00F</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>Reserved</td>
</tr>
<tr>
<td>RX_FIFO_DATA</td>
<td>0x010</td>
<td>RW</td>
<td>8</td>
<td>-</td>
<td>SPI block reading data register</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>[7:0] SD/MMC block read data, fifo size matches the SD/MMC block size of 512 bytes</td>
</tr>
<tr>
<td>RESERVED1</td>
<td>0x011</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>Reserved</td>
</tr>
<tr>
<td>RX_FIFO_DATA_COUNT_MSB</td>
<td>0x012</td>
<td>RO</td>
<td>8</td>
<td>-</td>
<td>MSB byte of reading data count register</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>[7:0] MSByte of FIFO_DATA_COUNT, indicates the number of data entries within the fifo</td>
</tr>
<tr>
<td>RX_FIFO_DATA_COUNT_LSB</td>
<td>0x013</td>
<td>RO</td>
<td>8</td>
<td>-</td>
<td>LSB byte of reading data count register</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>[7:0] LSByte of FIFO_DATA_COUNT, indicates the number of data entries within the fifo</td>
</tr>
<tr>
<td>RX_FIFO_CONTROL</td>
<td>0x014</td>
<td>WO</td>
<td>8</td>
<td>0x00</td>
<td>SD block reading data control register</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>[7:1] Reserved</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>[0] Force fifo empty</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1 = Force fifo empty, delete all the data samples within the fifo, self clearing</td>
</tr>
<tr>
<td>RESERVED2</td>
<td>0x015-0x019</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>Reserved</td>
</tr>
<tr>
<td>TX_FIFO_DATA</td>
<td>0x020</td>
<td>WO</td>
<td>8</td>
<td>-</td>
<td>SD block writing data register</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>[7:0] SD/MMC block write data, fifo size matches the SD/MMC block size of 512 bytes</td>
</tr>
<tr>
<td>RESERVED3</td>
<td>0x021-0x023</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>Reserved</td>
</tr>
<tr>
<td>TX_FIFO_CONTROL</td>
<td>0x024</td>
<td>WO</td>
<td>8</td>
<td>0x00</td>
<td>SD block writing data control register</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>[7:1] Reserved</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>[0] Force fifo empty</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1 = Force fifo empty, delete all the data samples within the fifo, self clearing</td>
</tr>
</tbody>
</table>
13.3 Drivers Usage

The usage of SD-Card drivers is as described in Table 13-2.

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SD_Init</td>
<td>SD hardware initialization</td>
</tr>
<tr>
<td>SD_BlockWrite</td>
<td>SD block write data, block size is 512 bytes.</td>
</tr>
<tr>
<td>SD_BlockRead</td>
<td>SD block read data, block size is 512 bytes.</td>
</tr>
</tbody>
</table>

13.4 Reference Design

Gowin_EMPU_M1 supports SD-Card reference design in ARM Keil MDK (V5.24 and above) and GOWIN MCU Designer (V1.1 and above) software environment. Get following reference designs by this link:

- Gowin_EMPU_M1\ref_design\MCU_RefDesign\Keil_RefDesign\fatfs
- Gowin_EMPU_M1\ref_design\MCU_RefDesign\GMD_RefDesign\cm1\fatfs
14 Controller Area Network

14.1 Features

Gowin_EMPU_M1 includes one CAN module accessed by AHB:

- AHB bus interface
- Compliant with CAN2.0A, CAN2.0B, and ISO 11898-1 standard
- CAN FD protocol
- Independent system clock and CAN bus clock
- Flexible shared buffer solution to achieve optimal buffer size to store, send, and receive messages in given applications
- 1-16 receiving filter can be configured
- Programmable baud rate prescaler

14.2 Registers

The definition of CAN registers is as described in Table 14-1.

<table>
<thead>
<tr>
<th>Register Name</th>
<th>Address Offset</th>
<th>Type</th>
<th>Width</th>
<th>Initial Value</th>
<th>Description</th>
</tr>
</thead>
</table>
| SRST          | 0x0000         | RW   | 32    | 0x00000000    | Software reset register
|               |                |      |       |               | [31:1] Reserved
|               |                |      |       |               | [0] control reset
|               |                |      |       |               | 1 = start hard reset
|               |                |      |       |               | 0 = cancel reset
| CMD           | 0x0004         | RW   | 32    | 0x00000000    | Command register
|               |                |      |       |               | [31:1] Reserved
|               |                |      |       |               | [0] Enable
|               |                |      |       |               | 1 = working mode
|               |                |      |       |               | 0 = command mode
| BRP           | 0x0008         | RW   | 32    | 0x00000000    | Baud rate prescalar register
|               |                |      |       |               | [31:8] Reserved
|               |                |      |       |               | [7:0] baud rate prescalar
| BTN           | 0x000C         | RW   | 32    | 0x00000000    | Bit timing (nominal) register
|               |                |      |       |               | [28:24] sjw_nom
<table>
<thead>
<tr>
<th>Register Name</th>
<th>Address Offset</th>
<th>Type</th>
<th>Width</th>
<th>Initial Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>PHASE_SEG2 window’s width [5:0] phseg1_nom, PROP_SEG+PHASE_SEG1 window’s width</td>
</tr>
<tr>
<td>BTD</td>
<td>0x0010</td>
<td>RW</td>
<td>32</td>
<td>0x00000000</td>
<td>Bit timing (data) register [26:24] sjw_d [11:8] phseg2_d, PHASE_SEG2 window’s depth [3:0] phseg1_d, PROP_SEG+PHASE_SEG1 window’s depth</td>
</tr>
<tr>
<td>RVD0</td>
<td>0x001C</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>Reserved</td>
</tr>
<tr>
<td>IC</td>
<td>0x0028</td>
<td>WO</td>
<td>32</td>
<td>0x00000000</td>
<td>Interrupt clear register [31] Clear bus off status [27] Clear TX message</td>
</tr>
<tr>
<td>Register Name</td>
<td>Address Offset</td>
<td>Type</td>
<td>Width</td>
<td>Initial Value</td>
<td>Description</td>
</tr>
<tr>
<td>---------------</td>
<td>----------------</td>
<td>------</td>
<td>-------</td>
<td>---------------</td>
<td>-------------</td>
</tr>
<tr>
<td>RSVD1</td>
<td>0x002C</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>Reserved</td>
</tr>
<tr>
<td>CFG</td>
<td>0x0030</td>
<td>RW</td>
<td>32</td>
<td>0x0000000000</td>
<td>Configuration register</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>[4] Configure disprotexceonres 1 = 'res' is FORM-ERROR 0 = 'res' is exception</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>[0] Configure isofd 1 = ISO FD mode 0 = non ISO FD mode</td>
</tr>
<tr>
<td>RSVD2</td>
<td>0x0034-0x003C</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>Reserved</td>
</tr>
<tr>
<td>RXBCFG</td>
<td>0x0040</td>
<td>RW</td>
<td>32</td>
<td>0x0000000000</td>
<td>RX buffer/fifo configuration register</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>[31:16] RX buffer’s ending offset</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>[15:0] RX buffer’s start offset</td>
</tr>
<tr>
<td>TXBCFG</td>
<td>0x0044</td>
<td>RW</td>
<td>32</td>
<td>0x0000000000</td>
<td>TX buffer/fifo configuration register</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>[31:16] TX buffer’s ending offset</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>[15:0] TX buffer’s start offset</td>
</tr>
<tr>
<td>TXHBCFG</td>
<td>0x0048</td>
<td>RW</td>
<td>32</td>
<td>0x0000000000</td>
<td>TX high-priority/fifo configuration register</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>[31:16] TX high-priority buffer’s ending offset</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>[15:0] TX high-priority buffer’s start offset</td>
</tr>
<tr>
<td>RSVD3</td>
<td>0x004C</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>Reserved</td>
</tr>
<tr>
<td>TXBRETRY</td>
<td>0x0050</td>
<td>RW</td>
<td>32</td>
<td>0x0000000000</td>
<td>TX buffer retry counter</td>
</tr>
<tr>
<td>TXHBRETRY</td>
<td>0x0054</td>
<td>RW</td>
<td>32</td>
<td>0x0000000000</td>
<td>TX high-priority buffer retry counter</td>
</tr>
<tr>
<td>TXMSGSTS</td>
<td>0x0058</td>
<td>RO</td>
<td>32</td>
<td>0x0000000000</td>
<td>Transmit message status register</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>[31:30] TX message status 00 = successfully</td>
</tr>
<tr>
<td>Register Name</td>
<td>Address Offset</td>
<td>Type</td>
<td>Width</td>
<td>Initial Value</td>
<td>Description</td>
</tr>
<tr>
<td>---------------</td>
<td>----------------</td>
<td>------</td>
<td>-------</td>
<td>---------------</td>
<td>-------------</td>
</tr>
<tr>
<td>TXHMSGSTS</td>
<td>0x005C</td>
<td>RO</td>
<td>32</td>
<td>0x00000000</td>
<td>Transmit high-priority message status register</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>[31:30] TX high-priority message status</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>00 = successfully</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>10 = retry</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>11 = failed</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>[28:0] Message ID</td>
</tr>
<tr>
<td>ERRSTS</td>
<td>0x0060</td>
<td>RW</td>
<td>32</td>
<td>0x00000000</td>
<td>Error status register</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>[4] CRC error</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>[3] ACK error</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>[2] FORM error</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>[1] BIT error</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>[0] STUFF error</td>
</tr>
<tr>
<td>ERRCNTR</td>
<td>0x0064</td>
<td>RO</td>
<td>32</td>
<td>0x00000000</td>
<td>Error counter register</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>[24:16] TX error counter</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>[8:0] RX error counter</td>
</tr>
<tr>
<td>RSVD4</td>
<td>0x0068-0x00fc</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>Reserved</td>
</tr>
<tr>
<td>AF</td>
<td>0x0100-0x100+</td>
<td>RW</td>
<td>32</td>
<td>0x00000000</td>
<td>Receive acceptance filter register</td>
</tr>
<tr>
<td></td>
<td>(4*N)</td>
<td></td>
<td></td>
<td></td>
<td>[31] Enable</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1 = valid</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0 = invalid</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>[30] IDE</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1 = extended frame</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0 = normal frame</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>[29] Extended data length</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1 = match FD frame</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0 = match normal frame</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>[28:18] Basic ID</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>[17:0] ID Extension</td>
</tr>
<tr>
<td>AFM</td>
<td>0x0140-0x140+</td>
<td>RW</td>
<td>32</td>
<td>0x00000000</td>
<td>Receive acceptance filter mask register</td>
</tr>
<tr>
<td></td>
<td>(4*N)</td>
<td></td>
<td></td>
<td></td>
<td>[28:18] Basic ID mask</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>[17:0] ID Extension mask</td>
</tr>
<tr>
<td>RSVD5</td>
<td>0x0180-0x01F</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>Reserved</td>
</tr>
<tr>
<td>RXB</td>
<td>0x0200</td>
<td>RO</td>
<td>32</td>
<td>0x00000000</td>
<td>Receive buffer/fifo window register</td>
</tr>
<tr>
<td>TXB</td>
<td>0x0204</td>
<td>WO</td>
<td>32</td>
<td>0x00000000</td>
<td>Transmit buffer/fifo window register</td>
</tr>
<tr>
<td>TXHB</td>
<td>0x0208</td>
<td>WO</td>
<td>32</td>
<td>0x00000000</td>
<td>Transmit high-priority buffer/fifo window register</td>
</tr>
<tr>
<td>TXBSTS</td>
<td>0x020C</td>
<td>RO</td>
<td>32</td>
<td>0x00000000</td>
<td>Transmit buffer/fifo status</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>[31] txbwerr</td>
</tr>
</tbody>
</table>
### 14.3 Driver Usage

The usage of CAN drivers is as described in Table 14-2.

#### Table 14-2 Usage of CAN Drivers

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>can_srst</td>
<td>Start hard reset</td>
</tr>
<tr>
<td>can_set_cmd</td>
<td>Enable working mode</td>
</tr>
<tr>
<td>can_set_brp</td>
<td>Set baud rate prescalar</td>
</tr>
<tr>
<td>can_set_btn_phseg1_nom</td>
<td>Set PROP_SEG+PHASE_SEG1 window’s width</td>
</tr>
<tr>
<td>can_set_btn_phseg2_nom</td>
<td>Set PHASE_SEG2 window’s width</td>
</tr>
<tr>
<td>can_set_btn_sjw_nom</td>
<td>Set sjw_nom</td>
</tr>
<tr>
<td>can_set_btn</td>
<td>Set BTN register</td>
</tr>
<tr>
<td>can_read_btn_phseg1_nom</td>
<td>Get PROP_SEG+PHASE_SEG1 window’s width</td>
</tr>
<tr>
<td>can_read_btn_phseg2_nom</td>
<td>Get PHASE_SEG2 window’s width</td>
</tr>
<tr>
<td>can_read_btn_sjw_nom</td>
<td>Get sjw_nom</td>
</tr>
<tr>
<td>can_set_btd_phseg1_d</td>
<td>Set PROP_SEG+PHASE_SEG1 window’s depth</td>
</tr>
<tr>
<td>can_set_btd_phseg2_d</td>
<td>Set PHASE_SEG2 window’s depth</td>
</tr>
<tr>
<td>can_set_btd_sjw_d</td>
<td>Set sjw_d</td>
</tr>
<tr>
<td>can_set_btd</td>
<td>Set BTD register</td>
</tr>
<tr>
<td>can_read_btd_phseg1_d</td>
<td>Get PROP_SEG+PHASE_SEG1 window’s depth</td>
</tr>
<tr>
<td>can_read_btd_phseg2_d</td>
<td>Get PHASE_SEG2 window’s depth</td>
</tr>
<tr>
<td>can_read_btd_sjw_d</td>
<td>Get sjw_d</td>
</tr>
<tr>
<td>can_read_is_bit</td>
<td>Get IS register bits function</td>
</tr>
<tr>
<td>can_set_ie_bit</td>
<td>Set IE register bits function</td>
</tr>
<tr>
<td>can_clear_ie_bit</td>
<td>Clear IE register bits function</td>
</tr>
<tr>
<td>can_read_ie_bit</td>
<td>Get IE register bits function</td>
</tr>
<tr>
<td>can_set_ic_bit</td>
<td>Set IC register bits function</td>
</tr>
<tr>
<td>can_set_cfg_bit_as_one</td>
<td>Set CFG register bits function</td>
</tr>
<tr>
<td>can_set_cfg_bit_as_zero</td>
<td>Clear CFG register bits function</td>
</tr>
<tr>
<td>can_read_cfg_bit</td>
<td>Get CFG register bits function</td>
</tr>
<tr>
<td>can_set_rxbcfg_rxb_start</td>
<td>Set RX buffer start offset in RXBCFG</td>
</tr>
<tr>
<td>can_read_rxbcfg_rxb_start</td>
<td>Get RX buffer start offset in RXBCFG</td>
</tr>
<tr>
<td>Name</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------------------------------</td>
<td>-------------------------------------------------------</td>
</tr>
<tr>
<td>can_set_rxbcfg_rxb_end</td>
<td>Set RX buffer ending offset in RXBCFG</td>
</tr>
<tr>
<td>can_read_rxbcfg_rxb_end</td>
<td>Get RX buffer ending offset in RXBCFG</td>
</tr>
<tr>
<td>set_rxbcfg</td>
<td>Set RX buffer start and ending offset</td>
</tr>
<tr>
<td>can_set_txbcfg_txb_start</td>
<td>Set TX buffer start offset in TXBCFG</td>
</tr>
<tr>
<td>can_read_txbcfg_txb_start</td>
<td>Get TX buffer start offset in TXBCFG</td>
</tr>
<tr>
<td>can_set_txbcfg_txb_end</td>
<td>Set TX buffer ending offset in TXBCFG</td>
</tr>
<tr>
<td>can_read_txbcfg_txb_end</td>
<td>Get TX buffer ending offset in TXBCFG</td>
</tr>
<tr>
<td>set_txbcfg</td>
<td>Set TX buffer start and ending offset</td>
</tr>
<tr>
<td>can_set_txhbcfg_txhb_start</td>
<td>Set TX high-priority buffer start offset in TXHBCFG</td>
</tr>
<tr>
<td>can_read_txhbcfg_txhb_start</td>
<td>Get TX high-priority buffer start offset in TXHBCFG</td>
</tr>
<tr>
<td>can_set_txhbcfg_txhb_end</td>
<td>Set TX high-priority buffer ending offset in TXHBCFG</td>
</tr>
<tr>
<td>can_read_txhbcfg_txhb_end</td>
<td>Get TX high-priority buffer ending offset in TXHBCFG</td>
</tr>
<tr>
<td>set_txhbcfg</td>
<td>Set TX high-priority buffer start and ending offset</td>
</tr>
<tr>
<td>can_set_txbrety</td>
<td>Set TX buffer retry</td>
</tr>
<tr>
<td>can_read_txbrety</td>
<td>Get TX buffer retry</td>
</tr>
<tr>
<td>can_set_txbrety</td>
<td>Set TX high-priority buffer retry counter</td>
</tr>
<tr>
<td>can_read_txbrety</td>
<td>Get TX high-priority buffer retry counter</td>
</tr>
<tr>
<td>can_read_txmsgsts</td>
<td>Get TX message status</td>
</tr>
<tr>
<td>can_read_txmsgid</td>
<td>Get TX message ID</td>
</tr>
<tr>
<td>can_read_txhmsgsts</td>
<td>Get TX high-priority message status</td>
</tr>
<tr>
<td>can_read_txhmsgid</td>
<td>Get TX high-priority message ID</td>
</tr>
<tr>
<td>can_read_errsts</td>
<td>Get error status</td>
</tr>
<tr>
<td>can_read_errcntr_rec</td>
<td>Get RX error counter</td>
</tr>
<tr>
<td>can_read_errcntr_tec</td>
<td>Get TX error counter</td>
</tr>
<tr>
<td>can_set_af_bit_as_one</td>
<td>Set AF bits function</td>
</tr>
<tr>
<td>can_set_af_bit_as_zero</td>
<td>Clear AF bits function</td>
</tr>
<tr>
<td>can_read_af_bit</td>
<td>Get AF bits function</td>
</tr>
<tr>
<td>can_set_af_ie</td>
<td>Set AF ID Extension</td>
</tr>
<tr>
<td>can_read_af_ie</td>
<td>Get AF ID Extension</td>
</tr>
<tr>
<td>can_set_af_bid</td>
<td>Set AF basic ID</td>
</tr>
<tr>
<td>can_read_af_bid</td>
<td>Get AF basic ID</td>
</tr>
<tr>
<td>can_set_afm_iem</td>
<td>Set AFM ID Extension mask</td>
</tr>
<tr>
<td>can_read_afm_iem</td>
<td>Get AFM ID Extension mask</td>
</tr>
<tr>
<td>can_set_afm_bidm</td>
<td>Set AFM basic ID mask</td>
</tr>
<tr>
<td>can_read_afm_bidm</td>
<td>Get AFM basic ID mask</td>
</tr>
<tr>
<td>can_read_rxb</td>
<td>Get RXB register</td>
</tr>
<tr>
<td>can_set_txb</td>
<td>Set TXB register</td>
</tr>
<tr>
<td>can_set_txhb</td>
<td>Set TXHB register</td>
</tr>
</tbody>
</table>

**IPUG533-1.6E**
<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>can_read_txbsts_tdbspace</td>
<td>Get txbspace</td>
</tr>
<tr>
<td>can_read_txbsts_txbwerr</td>
<td>Get txbwerr</td>
</tr>
<tr>
<td>can_read_txhbsts_tdbspace</td>
<td>Get txhbspace</td>
</tr>
<tr>
<td>can_read_txhbsts_txbwerr</td>
<td>Get txhbwerr</td>
</tr>
<tr>
<td>can_read_rxbsts</td>
<td>Get rxbdepth</td>
</tr>
</tbody>
</table>

### 14.4 Reference Design

Gowin_EMPU_M1 supports CAN reference design in ARM Keil MDK (V5.24 and above) and GOWIN MCU Designer (V1.1 and above) software environment. Get following reference designs by this [link]:

- Gowin_EMPU_M1\ref_design\MCU_RefDesign\Keil_RefDesign\can
- Gowin_EMPU_M1\ref_design\MCU_RefDesign\GMD_RefDesign\cm1_can
15 Ethernet

15.1 Features

Gowin_EMPU_M1 includes one Ethernet module accessed by AHB:

- AHB bus interface
- Realizes the function description of MAC layer in the IEEE802.3 protocol
- RGMII/GMII/MII interface
- Supports 10/100/1000M rate
- Supports full duplex and half duplex mode, and conflict detection can be supported in half duplex mode
- Supports users to choose whether to automatically add and verify CRC
- Supports to add pad function automatically
- Supports Ethernet frame classification statistics
- Supports Ethernet frame error statistics
- Supports IFG configurable functions
- Supports Jumbo mode
- Supports Flow Control in full duplex mode
- Supports Management interface mdc, mdio
## 15.2 Registers

The definition of Ethernet registers is as described in Table 15-1.

<table>
<thead>
<tr>
<th>Name</th>
<th>Address Offset</th>
<th>Type</th>
<th>Width</th>
<th>Initial Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ETH_TX_DATA</td>
<td>0x000-0x5FF</td>
<td>WO</td>
<td>32</td>
<td>0x00000000</td>
<td>Transmit data registers</td>
</tr>
<tr>
<td>ETH_RX_DATA</td>
<td>0x000-0x5FF</td>
<td>RO</td>
<td>32</td>
<td>0x00000000</td>
<td>Receive data registers</td>
</tr>
<tr>
<td>ETH_TX_LENGTH</td>
<td>0x600</td>
<td>RW</td>
<td>32</td>
<td>0x00000000</td>
<td>Transmit data length</td>
</tr>
<tr>
<td>ETH_TX_EN</td>
<td>0x604</td>
<td>RW</td>
<td>32</td>
<td>0x00000000</td>
<td>Transmit enable</td>
</tr>
<tr>
<td>ETH_TX_FAIL</td>
<td>0x608</td>
<td>RW</td>
<td>32</td>
<td>0x00000000</td>
<td>Transmit failed status</td>
</tr>
<tr>
<td>ETH_TX_IS</td>
<td>0x60C</td>
<td>RO</td>
<td>32</td>
<td>0x00000000</td>
<td>Transmit interrupt status</td>
</tr>
<tr>
<td>ETH_TX_IC</td>
<td>0x610</td>
<td>WO</td>
<td>32</td>
<td>0x00000000</td>
<td>Transmit interrupt clear</td>
</tr>
<tr>
<td>ETH_TX_IE</td>
<td>0x614</td>
<td>RW</td>
<td>32</td>
<td>0x00000000</td>
<td>Transmit interrupt enable</td>
</tr>
<tr>
<td>RESERVED_1</td>
<td>0x618-0x67F</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>Reserved</td>
</tr>
<tr>
<td>ETH_RX_LENGTH</td>
<td>0x680</td>
<td>RO</td>
<td>32</td>
<td>0x00000000</td>
<td>Receive data length</td>
</tr>
<tr>
<td>ETH_RX_IS</td>
<td>0x684</td>
<td>RO</td>
<td>32</td>
<td>0x00000000</td>
<td>Receive interrupt status</td>
</tr>
<tr>
<td>ETH_RX_IE</td>
<td>0x688</td>
<td>RW</td>
<td>32</td>
<td>0x00000000</td>
<td>Receive interrupt enable</td>
</tr>
<tr>
<td>ETH_RX_IC</td>
<td>0x68C</td>
<td>WO</td>
<td>32</td>
<td>0x00000000</td>
<td>Receive interrupt clear</td>
</tr>
<tr>
<td>RESERVED_2</td>
<td>0x690-0x6FF</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>Reserved</td>
</tr>
<tr>
<td>MIIM_OP_MODE</td>
<td>0x700</td>
<td>RW</td>
<td>32</td>
<td>0x00000000</td>
<td>MIIM operation mode</td>
</tr>
<tr>
<td>Name</td>
<td>Address Offset</td>
<td>Type</td>
<td>Width</td>
<td>Initial Value</td>
<td>Description</td>
</tr>
<tr>
<td>--------------------</td>
<td>----------------</td>
<td>------</td>
<td>-------</td>
<td>---------------</td>
<td>------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>MIIM_PHY_ADDR</td>
<td>0x704</td>
<td>RW</td>
<td>32</td>
<td>0x00000000</td>
<td>MIIM PHY address [31:5] Reserved [4:0] MIIM PHY address</td>
</tr>
<tr>
<td>MIIM_REG_ADDR</td>
<td>0x708</td>
<td>RW</td>
<td>32</td>
<td>0x00000000</td>
<td>MIIM reg address [31:5] Reserved [4:0] MIIM reg address</td>
</tr>
<tr>
<td>MIIM_WR_DATA</td>
<td>0x70C</td>
<td>RW</td>
<td>32</td>
<td>0x00000000</td>
<td>MIIM write data [31:16] Reserved [15:0] MIIM write data</td>
</tr>
<tr>
<td>MIIM_RD_DATA</td>
<td>0x710</td>
<td>RO</td>
<td>32</td>
<td>0x00000000</td>
<td>MIIM read data [31:16] Reserved [15:0] MIIM read data</td>
</tr>
<tr>
<td>MIIM_IS</td>
<td>0x714</td>
<td>RO</td>
<td>32</td>
<td>0x00000000</td>
<td>MIIM interrupt status [31:2] Reserved [1] MIIM operation end [0] MIIM read data valid</td>
</tr>
<tr>
<td>MIIM_IE</td>
<td>0x718</td>
<td>RW</td>
<td>32</td>
<td>0x00000000</td>
<td>MIIM interrupt enable [31:2] Reserved [1] MIIM operation end [0] MIIM read data valid</td>
</tr>
<tr>
<td>MIIM_IC</td>
<td>0x71C</td>
<td>WO</td>
<td>32</td>
<td>0x00000000</td>
<td>MIIM interrupt clear [31:2] Reserved [1] MIIM operation end [0] MIIM read data valid</td>
</tr>
<tr>
<td>MIIM_OPE_N</td>
<td>0x720</td>
<td>RW</td>
<td>32</td>
<td>0x00000000</td>
<td>MIIM operation enable [31:1] Reserved [0] Enable MIIM operation</td>
</tr>
<tr>
<td>ETH_MODE</td>
<td>0x724</td>
<td>RW</td>
<td>32</td>
<td>0x00000000</td>
<td>Ethernet operation mode [31:3] Reserved [2:0] duplex mode and speed 000 = full duplex 100M 001 = full duplex 1000M 010 = full duplex 10M 100 = half duplex 100M 110 = half duplex 10M</td>
</tr>
</tbody>
</table>

**Note:**
- **RW**: Read-Write
- **RO**: Read-Only
- **WO**: Write-Only

**Address Offset:**
- **0x704**: Start address for MIIM_PHY_ADDR
- **0x708**: Start address for MIIM_REG_ADDR
- **0x70C**: Start address for MIIM_WR_DATA
- **0x710**: Start address for MIIM_RD_DATA
- **0x714**: Start address for MIIM_IS
- **0x718**: Start address for MIIM_IE
- **0x71C**: Start address for MIIM_IC
- **0x720**: Start address for MIIM_OPE_N
- **0x724**: Start address for ETH_MODE

**Initial Value:**
- The initial values shown correspond to the default state for each register.

**Description:**
- The descriptions provide details about the functionality and settings of each register, including their address offsets and type (RW, RO, WO).

**Example:**
- MIIM_PHY_ADDR: This register is used to store the MIIM PHY address, which is crucial for identifying and configuring the PHY layer of an Ethernet device. The address offset is 0x704, and it is a 32-bit register that can be read and written (RW) with an initial value of 0x00000000.
15.3 Drivers Usage

The usage of Ethernet drivers is as described in Table 15-2.

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>eth_init</td>
<td>Initialize Ethernet</td>
</tr>
<tr>
<td>tx_int_event</td>
<td>TX interrupt</td>
</tr>
<tr>
<td>rx_int_event</td>
<td>RX interrupt</td>
</tr>
<tr>
<td>eth_tx</td>
<td>Ethernet TX</td>
</tr>
<tr>
<td>eth_set_mode</td>
<td>Set Ethernet duplex mode and speed</td>
</tr>
<tr>
<td>miim_wr_int_event</td>
<td>MIIM interface transmits interrupt</td>
</tr>
<tr>
<td>miim_rd_int_event</td>
<td>MIIM interface receives interrupt</td>
</tr>
<tr>
<td>miim_write</td>
<td>MIIM interface transmits data</td>
</tr>
<tr>
<td>miim_receive</td>
<td>MIIM interface receives data</td>
</tr>
</tbody>
</table>

15.4 Reference Design

Gowin_EMPU_M1 supports Ethernet reference design in ARM Keil MDK (V5.24 and above) and GOWIN MCU Designer (V1.1 and above) software environment. Get following reference designs by this [link]:

- Gowin_EMPU_M1\ref_design\MCU_RefDesign\Keil_RefDesign\ethernet
- Gowin_EMPU_M1\ref_design\MCU_RefDesign\GMD_RefDesign\cm1\ethernet
16 DDR3 Memory

16.1 Features

Gowin_EMPU_M1 includes one DDR3 Memory module accessed by AHB:

- AHB bus interface
- Compatible with DDR3 SDRAM device meeting industrial standard and the module compatible with JESD79-3F specification
- Supports memory data path width of 16 bits
- Supports UDIMM memory module
- Supports memory chip of x8 data width
- Programmable burst length 4
- Supports clock ratio 1:2 mode
16.2 Registers

The definition of DDR3 registers is as shown in Table 16-1.

<table>
<thead>
<tr>
<th>Name</th>
<th>Address Offset</th>
<th>Type</th>
<th>Width</th>
<th>Initial Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RESERVED</td>
<td>0x0000</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>Reserved</td>
</tr>
<tr>
<td>WR_ADDR</td>
<td>0x0004</td>
<td>RW</td>
<td>32</td>
<td>0x0</td>
<td>Write address register</td>
</tr>
<tr>
<td>WR_DATA</td>
<td>0x0008-0x0014</td>
<td>WO</td>
<td>128</td>
<td>0x0</td>
<td>Write data register</td>
</tr>
<tr>
<td>RD_ADDR</td>
<td>0x0018</td>
<td>RW</td>
<td>32</td>
<td>0x0</td>
<td>Read address register</td>
</tr>
<tr>
<td>RD_EN</td>
<td>0x001c</td>
<td>RW</td>
<td>32</td>
<td>0x0</td>
<td>Read enable register [31:1] Reserved [0] Read enable [1 = Enable 0 = Disable]</td>
</tr>
<tr>
<td>RD_DATA</td>
<td>0x0020-0x002c</td>
<td>RO</td>
<td>128</td>
<td>0x0</td>
<td>Read data register</td>
</tr>
<tr>
<td>INIT</td>
<td>0x0030</td>
<td>RW</td>
<td>32</td>
<td>0x0</td>
<td>Initialized completely flag register [31:1] Reserved [0] Initialized completely flag</td>
</tr>
<tr>
<td>WR_EN</td>
<td>0x0034</td>
<td>RW</td>
<td>32</td>
<td>0x0</td>
<td>Write enable and ending flag register [31:1] Reserved [0] Write enable and ending flag [1 = enable 0 = ending]</td>
</tr>
</tbody>
</table>

16.3 Drivers Usage

The usage of DDR3 drivers is as described in Table 16-2.

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DDR3_Init</td>
<td>Initialize DDR3</td>
</tr>
<tr>
<td>DDR3_Read</td>
<td>Read data from DDR3</td>
</tr>
<tr>
<td>DDR3_Write</td>
<td>Write data into DDR3</td>
</tr>
</tbody>
</table>

16.4 Reference Design

Gowin_EMPU_M1 supports DDR3 reference design in ARM Keil MDK (V5.24 and above) and GOWIN MCU Designer (V1.1 and above) software environment. Get following reference designs by this link:

- Gowin_EMPU_M1\ref_design\MCU_RefDesign\Keil_RefDesign\ddr3
- Gowin_EMPU_M1\ref_design\MCU_RefDesign\GMD_RefDesign\cm1\_ddr3
17 SPI-Flash

17.1 Features

Gowin_EMPU_M1 includes one SPI-Flash module accessed via AHB bus.

- SPI-Flash download is at AHB bus interface;
- SPI-Flash Memory read, write, and erase functions are at APB bus interface;
- The default is Gowin development board On-board Winbond W25Q64BV chip.

17.2 Registers

The definition of SPI-Flash registers is as shown in Table 17-1.

Table 17-1 Definition of SPI-Flash Registers

<table>
<thead>
<tr>
<th>Name</th>
<th>Address Offset</th>
<th>Type</th>
<th>Width</th>
<th>Initial Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>IDREV</td>
<td>0x00</td>
<td>RO</td>
<td>32</td>
<td>0x02002000</td>
<td>ID and revision register</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>[31:8] ID number</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>[7:4] Major revision number</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>[3:0] Minor revision number</td>
</tr>
<tr>
<td>RESERVED0[3]</td>
<td>0x04-0x0C</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>Reserved</td>
</tr>
<tr>
<td>TRANSFMT</td>
<td>0x10</td>
<td>RW</td>
<td>32</td>
<td>0x00020780</td>
<td>SPI transfer format register</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>[31:18] Reserved</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>[17:16] Address length in bytes</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>00 = 1 byte</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>01 = 2 bytes</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>10 = 3 bytes</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>11 = 4 bytes</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>[15:13] Reserved</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>[12:8] Data length</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>[7] Enable data merge mode</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>[6:5] Reserved</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>[4] Bi-directional MOSI in single mode</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0 = MOSI is uni-directional signal</td>
</tr>
<tr>
<td>Name</td>
<td>Address Offset</td>
<td>Type</td>
<td>Width</td>
<td>Initial Value</td>
<td>Description</td>
</tr>
<tr>
<td>---------------</td>
<td>----------------</td>
<td>------</td>
<td>-------</td>
<td>---------------</td>
<td>--------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>DIRECTIO</td>
<td>0x14</td>
<td>RW</td>
<td>32</td>
<td>0x0</td>
<td>SPI direct IO control register</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>[31:25] Reserved</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>[24] Enable direct IO</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0 = Disable</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1 = Enable</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>[23:22] Reserved</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>[21] Output enable for SPI-Flash hold signal</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>[20] Output enable for SPI-Flash write protect signal</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>[19] Output enable for the SPI MISO signal</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>[18] Output enable for the SPI MOSI signal</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>[17] Output enable for SPI SCLK signal</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>[16] Output enable for SPI CS signal</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>[15:14] Reserved</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>[13] Output value for SPI-Flash hold signal</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>[12] Output value for SPI-Flash write protect signal</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>[10] Output value for SPI MOSI signal</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>[9] Output value for SPI SCLK signal</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>[8] Output value for SPI CS signal</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>[7:6] Reserved</td>
</tr>
</tbody>
</table>

1 = MOSI is bi-directional signal
[3] Transfer data with the least significant bit first
0 = Most significant bit first
1 = Least significant bit first
[2] SPI master/slave mode selection
0 = Master mode
1 = Slave mode
[1] SPI clock polarity
0 = SCLK is LOW in the idle states
1 = SCLK is HIGH in the idle states
[0] SPI clock phase
0 = Sampling data at odd SCLK edges
1 = Sampling data at even SCLK edges
### Registers

<table>
<thead>
<tr>
<th>Name</th>
<th>Address Offset</th>
<th>Type</th>
<th>Width</th>
<th>Initial Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>RESERVED1[2]</strong></td>
<td>0x18-0x1C</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>Reserved</td>
</tr>
</tbody>
</table>
| **TRANSCTRL** | 0x20           | RW   | 32    | 0x0           | SPI transfer control register<br> [31] Reserved<br> [30] SPI command phase enable<br> 0 = Disable the command phase<br> 1 = Enable the command phase (Master mode only)<br> [29] SPI address phase enable<br> 0 = Disable the address phase<br> 1 = Enable the address phase (Master mode only)<br> [28] SPI address phase format<br> 0 = Address phase is single mode<br> 1 = The format of the address phase is the same as the DualQuad data phase (Master mode only)<br> [27:24] Transfer mode<br> 0000 = Write and read at the same time<br> 0001 = Write only<br> 0010 = Read only<br> 0011 = Write, Read<br> 0100 = Read, Write<br> 0101 = Write, Dummy, Read<br> 0110 = Read, Dummy, Write<br> 0111 = None data<br> 1000 = Dummy, Write<br> 1001 = Dummy, Read<br> 1010-1111 = Reserved<br> [23:22] SPI data phase format<br> 00 = Single mode<br> 01 = Dual I/O mode<br> 10 = Quad I/O mode<br> 11 = Reserved<br> [21] Append and one-byte special token following the address phase for SPI read transfers<br> [20:12] Transfer count for write data<br> [11] The value of the one-byte special token following the
<table>
<thead>
<tr>
<th>Name</th>
<th>Address Offset</th>
<th>Type</th>
<th>Width</th>
<th>Initial Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>address phase for SPI read transfers</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0 = token value is 0x00</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1 = token value is 0x69</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>[10:9] Dummy data count</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>[8:0] Transfer count for read data</td>
</tr>
<tr>
<td>CMD</td>
<td>0x24</td>
<td>RW</td>
<td>32</td>
<td>0x0</td>
<td>SPI command register</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>[31:8] Reserved</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>[7:0] SPI command</td>
</tr>
<tr>
<td>ADDR</td>
<td>0x28</td>
<td>RW</td>
<td>32</td>
<td>0x0</td>
<td>SPI address register</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>[31:0] SPI address (Master mode only)</td>
</tr>
<tr>
<td>DATA</td>
<td>0x2C</td>
<td>RW</td>
<td>32</td>
<td>0x0</td>
<td>SPI data register</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>[31:0] Data to transmit or the received data</td>
</tr>
<tr>
<td>CTRL</td>
<td>0x30</td>
<td>RW</td>
<td>32</td>
<td>0x0</td>
<td>SPI controller register</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>[31:21] Reserved</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>[20:16] Transmit FIFO threshold</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>[15:13] Reserved</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>[12:8] Receive FIFO threshold</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>[7:5] Reserved</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>[4] TX DMA enable</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>[3] RX DMA enable</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>[2] Transmit FIFO reset</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>[1] Receive FIFO reset</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>[0] SPI reset</td>
</tr>
<tr>
<td>STATUS</td>
<td>0x34</td>
<td>RO</td>
<td>32</td>
<td>0x0</td>
<td>SPI status register</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>[31:24] Reserved</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>[23] Transmit FIFO full flag</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>[22] Transmit FIFO empty flag</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>[21] Reserved</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>[20:16] Number of valid entries int the transmit FIFO</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>[15] Receive FIFO full flag</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>[14] Receive FIFO empty flag</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>[13] Reserved</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>[12:8] Number of valid entries in the receive FIFO</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>[7:1] Reserved</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>[0] SPI register programming is in progress</td>
</tr>
<tr>
<td>INTREN</td>
<td>0x38</td>
<td>RW</td>
<td>32</td>
<td>0x0</td>
<td>SPI interrupt enable register</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>[31:6] Reserved</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>[5] Enable the slave command interrupt</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>[4] Enable the end of SPI transfer interrupt</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>[3] Enable the SPI transmit FIFO</td>
</tr>
<tr>
<td>Name</td>
<td>Address Offset</td>
<td>Type</td>
<td>Width</td>
<td>Initial Value</td>
<td>Description</td>
</tr>
<tr>
<td>----------------</td>
<td>----------------</td>
<td>------</td>
<td>-------</td>
<td>---------------</td>
<td>-----------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>INTRST</td>
<td>0x3C</td>
<td>WO</td>
<td>32</td>
<td>0x0</td>
<td>SPI interrupt status register</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>[31:6] Reserved</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>[5] Slave command interrupt (Slave mode only)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>[4] End of SPI transfer interrupt</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>[3] TX FIFO threshold interrupt</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>[2] RX FIFO threshold interrupt</td>
</tr>
<tr>
<td>TIMING</td>
<td>0x40</td>
<td>RW</td>
<td>32</td>
<td>0x0</td>
<td>SPI interface timing register</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>[31:14] Reserved</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>[13:12] The minimum time between the edges of SPI CS and the edges of SCLK</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>[11:8] The minimum time the SPI CS should stay HIGH</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>[7:0] The clock frequency ratio between the clock source and SPI interface SCLK</td>
</tr>
<tr>
<td>RESERVED2[3]</td>
<td>0x44-0x4c</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>Reserved</td>
</tr>
<tr>
<td>MEMCTRL</td>
<td>0x50</td>
<td>RW</td>
<td>32</td>
<td>0x0</td>
<td>SPI memory access control register</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>[31:9] Reserved</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>[8] This bit is set when “MEMCTRL” / “TIMING” is written</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>[7:4] Reserved</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>[3:0] Selects the SPI command</td>
</tr>
<tr>
<td>RESERVED3[3]</td>
<td>0x54-0x5C</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>Reserved</td>
</tr>
<tr>
<td>SLVST</td>
<td>0x60</td>
<td>RW</td>
<td>32</td>
<td>0x0</td>
<td>SPI slave status register</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>[31:19] Reserved</td>
</tr>
<tr>
<td>SLVDATACNT</td>
<td>0x64</td>
<td>RO</td>
<td>32</td>
<td>0x0</td>
<td>SPI slave data count register</td>
</tr>
</tbody>
</table>

threshold interrupt
[2] Enable the SPI receive FIFO threshold interrupt
[1] Enable SPI transmit FIFO underrun interrupt (Slave mode only)
[0] Enable SPI receive FIFO overrun interrupt (Slave mode only)
<table>
<thead>
<tr>
<th>Name</th>
<th>Address Offset</th>
<th>Type</th>
<th>Width</th>
<th>Initial Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RESERED4[5]</td>
<td>0x68-0x78</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>Reserved</td>
</tr>
<tr>
<td>CONFIG</td>
<td>0x7C</td>
<td>RO</td>
<td>32</td>
<td>0x0</td>
<td>Configuration register</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>[31:15] Reserved</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>[14] Support for SPI slave mode</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>[13] Reserved</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>[12] Support for memory-mapped access through AHB bus</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>[10] Reserved</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>[9] Support for Quad I/O SPI</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>[9] Support for Dual I/O SPI</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>[7:6] Reserved</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>[5:4] Depth of TX FIFO</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>00 = 2 words</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>01 = 4 words</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>10 = 8 words</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>11 = 16 words</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>[3:2] Reserved</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>[1:0] Depth of RX FIFO</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>00 = 2 words</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>01 = 4 words</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>10 = 8 words</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>11 = 16 words</td>
</tr>
</tbody>
</table>
17.3 Drivers Usage

The usage of SPI-Flash drivers is as described in Table 17-2.

Table 17-2 Usage of SPI-Flash Drivers

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>spi_flash_init</td>
<td>Initialize SPI-Flash</td>
</tr>
<tr>
<td>change_mode_spi_flash</td>
<td>Switch SPI-Flash mode between download and read, write, erase memory</td>
</tr>
<tr>
<td>spi_flash_read</td>
<td>Read data from SPI-Flash</td>
</tr>
<tr>
<td>spi_flash_write</td>
<td>Write data into SPI-Flash</td>
</tr>
<tr>
<td>spi_flash_write_read</td>
<td>Write data into SPI-Flash and read data from SPI-Flash once time</td>
</tr>
<tr>
<td>spi_flash_page_program</td>
<td>Write data into SPI-Flash with pages</td>
</tr>
<tr>
<td>spi_flash_sector_erase</td>
<td>Erase SPI-Flash with sector</td>
</tr>
</tbody>
</table>

17.4 Reference Design

Gowin_EMPU_M1 supports SPI-Flash reference design in ARM Keil MDK (V5.24 and above) and GOWIN MCU Designer (V1.1 and above) software environment. Get following reference designs by this link:

- Gowin_EMPU_M1\ref_design\MCU_RefDesign\Keil_RefDesign\spi_flash
- Gowin_EMPU_M1\ref_design\MCU_RefDesign\Keil_RefDesign\fatfs
- Gowin_EMPU_M1\ref_design\MCU_RefDesign\GMD_RefDesign\cm1_spi_flash
- Gowin_EMPU_M1\ref_design\MCU_RefDesign\GMD_RefDesign\cm1_fatfs
18 PSRAM

18.1 Features

Gowin_EMPU_M1 includes one PSRAM module accessed via AHB bus:

- AHB bus interface
- Compatible with standard PSRAM devices
- Supports memory data path width of 8
- Support chips of x8 data width
- Programmable burst length of 32
- Clock ratio 1:2
- Supports initial delay of 6
- Supports regular delay mode
- Supports power off option
- Supports driving strength of 50
- Self-refreshing is full
- Refreshing rate is normal
### 18.2 Registers

The definition of PSRAM registers is as shown in Table 18-1.

<table>
<thead>
<tr>
<th>Name</th>
<th>Address Offset</th>
<th>Type</th>
<th>Width</th>
<th>Initial Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CMD</td>
<td>0x00</td>
<td>RW</td>
<td>1</td>
<td>0x0</td>
<td>Command register [0] Operation type 0 = Read operation 1 = Write operation</td>
</tr>
<tr>
<td>ADDRESS</td>
<td>0x04</td>
<td>RW</td>
<td>21</td>
<td>0x0</td>
<td>Address register [20:0] Address of reading and writing data</td>
</tr>
<tr>
<td>WR_DATA0</td>
<td>0x08</td>
<td>RW</td>
<td>32</td>
<td>0x0</td>
<td>Write data register 0 [31:0] Write first 32bit data</td>
</tr>
<tr>
<td>WR_DATA1</td>
<td>0x0C</td>
<td>RW</td>
<td>32</td>
<td>0x0</td>
<td>Write data register 1 [31:0] Write second 32bit data</td>
</tr>
<tr>
<td>WR_DATA2</td>
<td>0x10</td>
<td>RW</td>
<td>32</td>
<td>0x0</td>
<td>Write data register 2 [31:0] Write third 32bit data</td>
</tr>
<tr>
<td>WR_DATA3</td>
<td>0x14</td>
<td>RW</td>
<td>32</td>
<td>–</td>
<td>Write data register 3 [31:0] Write fourth 32bit data</td>
</tr>
<tr>
<td>CMD_EN</td>
<td>0x18</td>
<td>WO</td>
<td>1</td>
<td>–</td>
<td>Command enable register [0] Enable PSRAM</td>
</tr>
<tr>
<td>READ_DONE</td>
<td>0x1C</td>
<td>RW</td>
<td>1</td>
<td>–</td>
<td>Read status register [0] Read done flag, auto set 1 if it is done, and need mcu to clear</td>
</tr>
<tr>
<td>RD_DATA0</td>
<td>0x20</td>
<td>RO</td>
<td>32</td>
<td>–</td>
<td>Read data register 0 [31:0] Read first 32bit data</td>
</tr>
<tr>
<td>RD_DATA1</td>
<td>0x24</td>
<td>RO</td>
<td>32</td>
<td>–</td>
<td>Read data register 1 [31:0] Read second 32bit data</td>
</tr>
<tr>
<td>RD_DATA2</td>
<td>0x28</td>
<td>RO</td>
<td>32</td>
<td>–</td>
<td>Read data register 2 [31:0] Read third 32bit data</td>
</tr>
<tr>
<td>RD_DATA3</td>
<td>0x2C</td>
<td>RO</td>
<td>32</td>
<td>–</td>
<td>Read data register 3 [31:0] Read fourth 32bit data</td>
</tr>
<tr>
<td>INTI_DONE</td>
<td>0x30</td>
<td>RO</td>
<td>1</td>
<td>–</td>
<td>Initialization done register [0] PSRAM hardware initialization done flag 0 = Initialization failed 1 = Initialization done</td>
</tr>
</tbody>
</table>
18.3 Drivers Usage

The usage of PSRAM drivers is as described in Table 18-2.

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PSRAM_Check_Init_Status</td>
<td>Check the status of PSRAM initialization</td>
</tr>
<tr>
<td>PSRAM_Mode_Set</td>
<td>Set the mode for PSRAM write and read</td>
</tr>
<tr>
<td>PSRAM_Address_Set</td>
<td>Set the address of PSRAM and save data into this address</td>
</tr>
<tr>
<td>PSRAM_Read_Data_Buff</td>
<td>Read data from the buffer of PSRAM</td>
</tr>
<tr>
<td>PSRAM_Cmd_Enable</td>
<td>Enable the command of PSRAM</td>
</tr>
<tr>
<td>PSRAM_Read_Done_Flag</td>
<td>Get the flag of read PSRAM done</td>
</tr>
<tr>
<td>PSRAM_Clear_Read_Done_Flag</td>
<td>Clear the flag of read PSRAM done</td>
</tr>
<tr>
<td>PSRAM_Write_Data_Buff</td>
<td>Write data into the buffer of PSRAM</td>
</tr>
<tr>
<td>PSRAM_Cmd_Unable</td>
<td>Disable the command of PSRAM</td>
</tr>
<tr>
<td>PSRAM_Write_Data_Package</td>
<td>Write a package data into PSRAM</td>
</tr>
<tr>
<td>PSRAM_Read_Data_Package</td>
<td>Read a package data from PSRAM</td>
</tr>
</tbody>
</table>

18.4 Reference Design

Gowin_EMPU_M1 supports PSRAM reference design in ARM Keil MDK (V5.24 and above) and GOWIN MCU Designer (V1.1 and above) software environment. Get following reference designs by this link:

- Gowin_EMPU_M1\ref_design\MCU_RefDesign\Keil_RefDesign\psram
- Gowin_EMPU_M1\ref_design\MCU_RefDesign\Keil_RefDesign\fatfs
- Gowin_EMPU_M1\ref_design\MCU_RefDesign\GMD_RefDesign\cm1_psram
- Gowin_EMPU_M1\ref_design\MCU_RefDesign\GMD_RefDesign\cm1_fatfs
Gowin_EMPU_M1 supports the embedded operating system of uC/OS-III and FreeRTOS.

### 19.1 uC/OS-III

#### 19.1.1 Features

- The uC/OS-III is an extensible, romable, and preemptive real-time core. There is no limit to the number of tasks that can be managed.
- uC/OS-III is the third generation core. It provides a real-time core's functions, including resource management, synchronization, and inter-task communication, etc.
- uC/OS-III also provides features that are not available for the other real-time cores. For example, it can measure operating performance during runtime and send signals or messages to tasks directly. Tasks can also wait for multiple semaphores and message queues simultaneously.
- Gowin_EMPU_M1 offers uC/OS-III reference designs.
- uC/OS-III source code is available at Micrium website: [http://www.micrium.com](http://www.micrium.com).

#### 19.1.2 Operation System Version

Gowin_EMPU_M1 reference design uses uC/ os-iii V3.03.00 version.

#### 19.1.3 Operation System Configuration

- UCOSIII_CONFIG\os_cfg.hn and os_cfg_app.h can be modified to configure uC/OS-III.
- UCOS_BSP\bsp.c and bsp.h can be modified to support the development board used.
19.1.4 Reference Design

Gowin_EMPU_M1 supports uC/OS-III reference designs in ARM Keil MDK (V5.24 and above) and GOWIN MCU Designer (V1.1 and above) software environment. Get following reference designs by this link:

- Gowin_EMPU_M1\ref_design\MCU_RefDesign\Keil_RefDesign\ucos_iii
- Gowin_EMPU_M1\ref_design\MCU_RefDesign\GMD_RefDesign\cm1_freertos

19.2 FreeRTOS

19.2.1 Features

- FreeRTOS is a mini real-time operating system.
- FreeRTOS is a lightweight operating system. It offers functions of task management, time management, semaphore, message queue, memory management, recording, software timer, coroutines, etc. It can basically meet the needs of small systems.
- FreeRTOS is a free operating system. It has features of open source, portability, reducibility, and flexible scheduling policy.
- Gowin_EMPU_M1 offers FreeRTOS reference designs.
- FreeRTOS source code is available at FreeRTOS website: http://www.FreeRTOS.org

19.2.2 Operation System Version

Gowin_EMPU_M1 reference design uses FreeRTOS V10.2.1.

19.2.3 Operation System Configuration

"include\FreeRTOSConfig.h" can be modified to configure FreeRTOS.

19.2.4 Reference Design

Gowin_EMPU_M1 supports FreeRTOS reference design in ARM Keil MDK (V5.24 and above) and GOWIN MCU Designer (V1.1 and above) software environment. Get following reference designs by this link:

- Gowin_EMPU_M1\ref_design\MCU_RefDesign\Keil_RefDesign\freeRTOS
- Gowin_EMPU_M1\ref_design\MCU_RefDesign\GMD_RefDesign\cm1_freertos