Gowin_EMPU_M1 Software Programming
Reference Manual

IPUG533-1.6E,06/12/2020

Disclaimer

Revision History

Date	Version	Description
02/19/2019	1.0E	Initial version published.
07/18/2019	1.1E	MCU hardware design and software programming design support extended peripherals: CAN, Ethernet, SPI-Flash, RTC, DualTimer, TRNG, I2C, SPI, SD-Card.
08/18/2019	1.2E	● MCU hardware design and software programming design support extended peripheral: DDR3 Memory; ● Known issues of ITCM, DTCM Size and IDE fixed.
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

List of Figures

List of Tables

1 Software Programming Library

Gowin_EMPU_M1 offers software programming library: Gowin_EMPU_M1\src\c_lib

Note!

The above library link is given at http://cdn.gowinsemi.com.cn/Gowin_EMPU_M1.zip

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 1-1 Cortex-M1 Core Software Programming

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

File	Description
GOWIN_M1_rtc.c	RTC driving function definition
GOWIN_M1_trng.c	TRNG driving function definition
GOWIN_M1_dualtimer.c	DualTimer driving function definition
GOWIN_M1_i2c.c	I2C Master driving function definition
GOWIN_M1_spi.c	SPI Master driving function definition
GOWIN_M1_sdcard.c	SD-Card driving function definition
GOWIN_M1_misc.c	Interrupt priority management and SysTick
GOWIN_M1_it.c	Definition of interrupt handling function
retarget.c	Retargeting of UART0 printf function
malloc.c	Dynamic memory management

1.2 Embedded Operating System Software Programming

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

- uC/OS-III
- FreeRTOS

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 2-1 Standard Peripheral Memory Mapping Definitions

Standard Peripheral	Type	Address Mapping	Description
ITCM	-	0x00000000	1KB, 2KB, 4KB, 8KB, 16KB, 32KB, 64KB, 128KB, 256KB
DTCM	-	0x20000000	1KB, 2KB, 4KB, 8KB, 16KB, 32KB, 64KB, 128KB, 256KB
External Instruction Memory	-	0x00000000	External instruction memory
External Data Memory	-	0x20100000	External data memory
TIMER0	TIMER_TypeDef	0x50000000	TIMER0
TIMER1	TIMER_TypeDef	0x50001000	Timer1
UART0	UART_TypeDef	0x50004000	UART0
UART1	UART_TypeDef	0x50005000	UART1
Watch Dog	WDOG_TypeDef	0x50008000	Watchdog
RTC	RTC_RegDef	0x50006000	Real-time clock
TRNG	TRNG_RegDef	0x5000F000	True Random Number Generator
DualTimer	DUALTIMER_RegDef	0x50002000	Dual Timer
SPI_FLASH	SPI_FLASH_RegDef	0x50003000	Serial Peripheral Interface Flash
I2C	I2C_TypeDef	0x5000A000	Internal Integrated Circuit Bus
SPI	SPI_TypeDef	0x5000B000	SPI
SD-Card	SDCard_TypeDef	0x50009000	SD
GPIO0	GPIO_TypeDef	0x40000000	GPIO port
CAN	CAN_RegDef	0x45000000	Controller Area Network

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 [2]	-	0x60100000	Extension of APB Master [2]
APB Master [3]	-	0x60200000	Extension of APB Master [3]
APB Master [4]	-	0x60300000	Extension of APB Master [4]
APB Master [5]	-	0x60400000	Extension of APB Master [5]
APB Master [6]	-	0x60500000	Extension of APB Master [6]
APB Master [7]	-	0x60600000	Extension of APB Master [7]
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 [11]	-	0x60A00000	Extension of APB Master [11]
APB Master [12]	-	0x60B00000	Extension of APB Master [12]
APB Master [13]	-	0x60C00000	Extension of APB Master [13]
APB Master [14]	-	0x60D00000	Extension of APB Master [14]
APB Master [15]	-	0x60E00000	Extension of APB Master [15]
APB Master [16]	-	0x60F00000	Extension of APB Master [16]
AHB Master [1]	-	0x80000000	Extension of AHB Master [1]
AHB Master [2]	-	0x81000000	Extension of AHB Master [2]
AHB Master [3]	-	0x86000000	Extension of AHB Master [3]
AHB Master [4]	-	0x89000000	Extension of AHB Master [4]
AHB Master [5]	-	0x8A000000	Extension of AHB Master [5]
AHB Master [6]	-	0x8B000000	Extension of AHB Master [6]

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

System Control	Type	Address Mapping	Description
SysTick	SysTick_Type	0xE000E010	SysTick configuration struct
NVIC	NVIC_BASE	0xE000E100	NVIC configuration struct
SCnSCB	SCnSCB_Type	0xE000E000	System control register not in SCB
SCB	SCB_Type	0xE000ED00	SCB configuration struct

3 Interrupt Handling

Gowin_EMPU_M1nested 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-M1interrupt controller is as shown in Table 3-1.

Table 3-1 Definition of Interrupt Controller

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

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

4 Universal Asynchronous Receiver/Transmitter

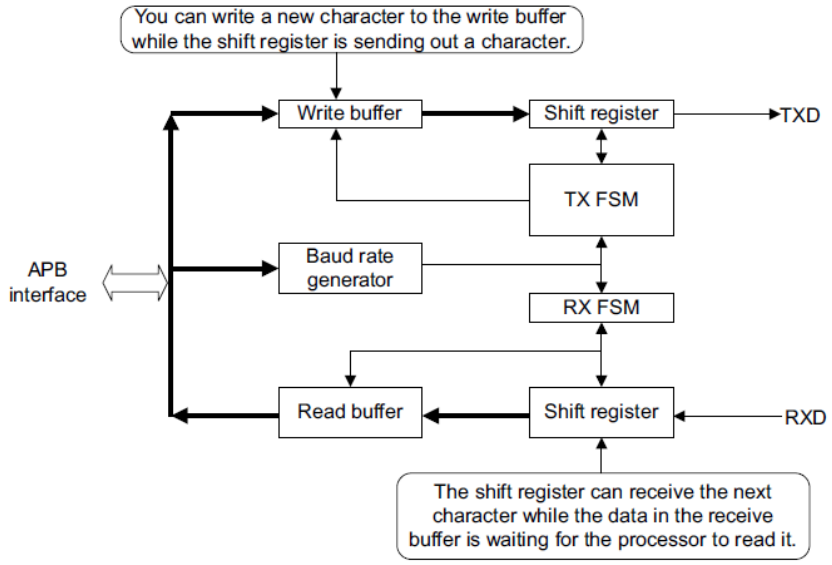
4.1 Features

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 4-1 Definition of UART Registers

Register Name	Address Offset	Type	Width	Initial Value	Description
DATA	0x000	RW	8	0x--	[7:0] Data Value
STATE	0x004	RW	4	0x0	[3] RX buffer overrun,write 1 to clear [2] TX buffer overrun,write 1 to clear [1] RX buffer full,read-only [0] TX buffer full,read-only
CTRL	0x008	RW	7	0x00	[6] High speed test mode for TX only [5] RX overrun interrupt enable [4] TX overrun interrupt enable [3] RX interrupt enable [2] TX interrupt enable [1] RX enable [0] TX enable
INTSTATUS/ INTCLEAR	0x00C	RW	4	0x0	[3] RX overrun interrupt,write 1 to clear [2] TX overrun interrupt,write 1 to clear [1] RX interrupt,write 1 to clear [0] TX interrupt,write 1 to clear
BAUDDIV	0x010	RW	20	0x00000	[19:0] Baud rate divider,the minimum number is 16

4.3 Initialization

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

Table 4-2 UART Initialization Definition

Name	Type	Value	Description
UART_BaudRate	uint32_t	Max 921.6Kbit/s	Baud rate
UART_Mode	UARTMode_TypeDef	ENABLE/DISABLE	Enable/Disable TX/RX mode
UART_Int	UARTInt_TypeDef	ENABLE/DISABLE	Enable/Disable TX/RX interrupt
UART_Ovr	UARTOvr_TypeDef	ENABLE/DISABLE	Enable/Disable TX/RX overrun interrupt
UART_Hstm	FunctionalState	ENABLE/DISABLE	Enable/Disable TX hisgh speed test mode

4.4 Drivers Usage

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

Table 4-3 Usage of UART Drivers

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

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

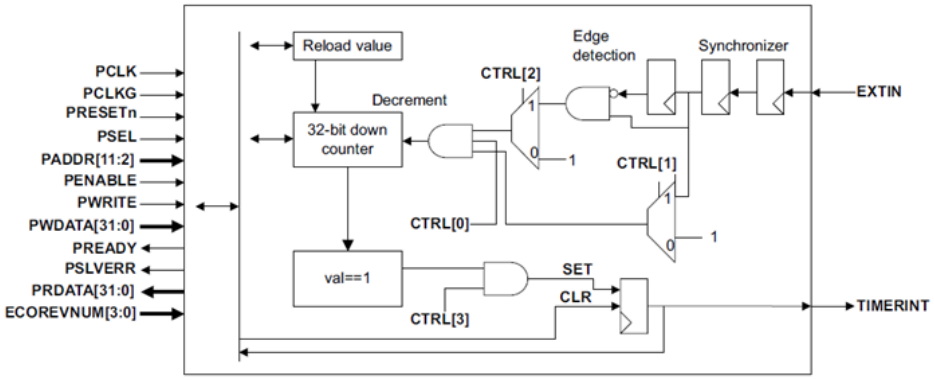
5.1 Features

Gowin_EMPU_M1 incluldes 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



5.2 Registers

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

Table 5-1 Definition of Timer Registers

Register Name	Address Offset	Type	Width	Initial Value	Description
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	0x00000000	[31:0] Current value
RELOAD	0x008	RW	32	0x00000000	[31:0] Reload value,writing to

Register Name	Address Offset	Type	Width	Initial Value	Description
					this register sets the current value
INTSTATUS/INTCLEAR	0x00C	RW	1	0x0	[0] Timer interrupt,write 1 to clear

5.3 Initialization

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

Table 5-2 Timer Initialization Definition

Name	Type	Value	Description
Reload	uint32_t	-	Reload value
TIMER_Int	TIMERInt_TypeDef	SET/RESET	Enable/Disable interrupt
TIMER_Exti	TIMERExti_TypeDef	-	External input as enable or clock

5.4 Drivers Usage

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

Table 5-3 Usage of Timer Drivers

Name	Description
TIMER_Init	Initialize TIMERx
TIMER_StartTimer	Start TIMERx
TIMER_StopTimer	Stop TIMERx
TIMER_GetIRQStatus	Return TIMERx interrupt status
TIMER_ClearIRQ	Clear TIMERx interrupt status
TIMER_GetReload	Return TIMERx reload value
TIMER_SetReload	Set TIMERx reload value
TIMER_GetValue	Return TIMERx current value
TIMER_SetValue	Set TIMERx current value
TIMER_EnableIRQ	Enable TIMERx interrupt request
TIMER_DisableIRQ	Disable TIMERx interrupt request

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

6 Watchdog

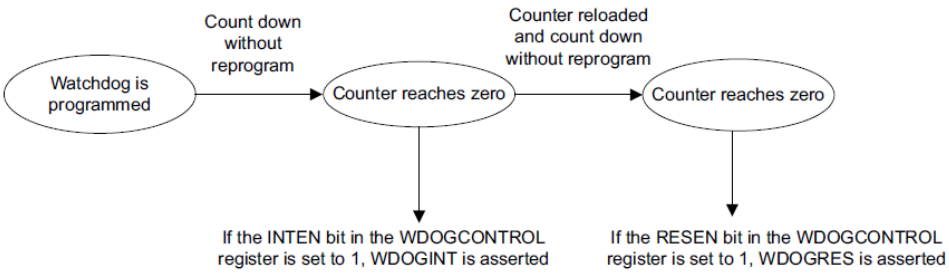
6.1 Features

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

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

6.3 Initialization

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

Table 6-2 Initialization of Watch Dogs

Name	Type	Value	Description
WDOG_Reload	uint32_t	-	Reload value
WDOG_Lock	WDOGLock_TypeDef	SET/RESET	Enable/Disable lock register write access
WDOG_Res	WDOGRes_TypeDef	SET/RESET	Enable/Disable reset flag
WDOG_Int	WDOGInt_TypeDef	SET/RESET	Enable/Disable interrupt flag
WDOG_ITMode	WDOGMode_Typedef	SET/RESET	Enable/Disable integration test mode flag

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

Name	Description
WDOG_Init	Initializes WatchDog
WDOG_RestartCounter	Restart watchdog counter
WDOG_GetCounterValue	Returns counter value
WDOG_SetResetEnable	Sets reset enable
WDOG_GetResStatus	Returns reset status
WDOG_SetIntEnable	Sets interrupt enable
WDOG_GetIntStatus	Returns interrupt enable
WDOG_ClrIntEnable	Clears interrupt enable
WDOG_GetRawIntStatus	Returns raw interrupt status
WDOG_GetMaskIntStatus	Returns masked interrupt status
WDOG_LockWriteAccess	Disable write access all registers
WDOG_UnlockWriteAccess	Enable write access all registers
WDOG_SetITModeEnable	Sets integration test mode enable
WDOG_ClrITModeEnable	Clears integration test mode enable
WDOG_GetITModeStatus	Returns integration test mode status
WDOG_SetITOP	Sets integration test output reset or interrupt
WDOG_GetITOPResStatus	Returns integration test output reset status
WDOG_GetITOPIntStatus	Returns integration test output interrupt status
WDOG_ClrITOP	Clears integration test output reset or interrupt

6.5 Reference Design

Gowin_EMPU_M1 supports Watch Dog reference design in ARM Keil MDK (V5.24and 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

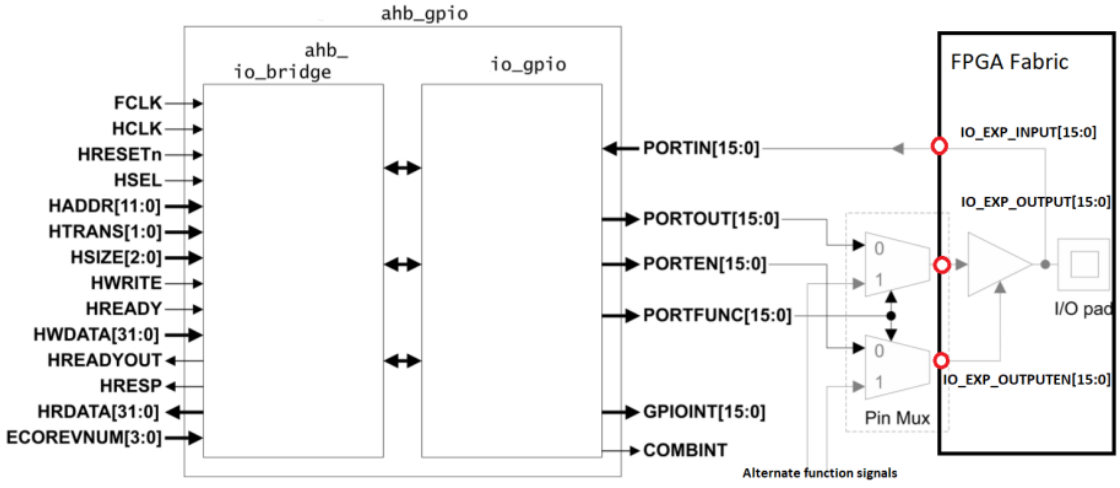
7.1 Features

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 7-1 Definition of GPIO Registers

Register Name	Address Offset	Type	Width	Initial Value	Description
DATA	0x0000	RW	16	0x----	[15:0] Data value Read Sampled at pin Write to data output register Read back value goes through double flip-flop

Register Name	Address Offset	Type	Width	Initial Value	Description
					synchronize logic with delay of two cycles
DATAOUT	0x0004	RW	16	0x0000	[15:0] Data output register value Read current value of data output register Write to data output register
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 intput 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 intput 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
INTENCLR	0x0024	RW	16	0x0000	[15:0] Interrupt enable

Register Name	Address Offset	Type	Width	Initial Value	Description
					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 /INTCLEAR	0x0038	RW	16	0x0000	[15:0] Write IRQ status clear register Write 1 to clear interrupt request Write 0 no effectx Read back IRQ status register

Register Name	Address Offset	Type	Width	Initial Value	Description
MASKLOWBYTE	0x0400 -0x07FC	RW	16	0x----	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
MASKHIGHBYTE	0x0800 -0x0BFC	RW	16	0x----	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
RESERVED	0x0C00 -0x0FCF	-	-	-	Reserved

7.3 Initialization

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

Table 7-2 GPIO Initialization

Name	Type	Value	Description
GPIO_Pin	uint32_t	GPIO_Pin_0 GPIO_Pin_1 GPIO_Pin_2 GPIO_Pin_3 GPIO_Pin_4 GPIO_Pin_5 GPIO_Pin_6 GPIO_Pin_7 GPIO_Pin_8 GPIO_Pin_9 GPIO_Pin_10 GPIO_Pin_11 GPIO_Pin_12 GPIO_Pin_13 GPIO_Pin_14 GPIO_Pin_15	16 bits GPIO Pins
GPIO_Mode	GPIOMode_TypeDef	GPIO_Mode_IN GPIO_Mode_OUT GPIO_Mode_AF	16 bits GPIO Pins mode
GPIO_Int	GPIOInt_TypeDef	GPIO_Int_Disable GPIO_Int_Low_Level GPIO_Int_High_Level GPIO_Int_Falling_Edge GPIO_Int_Rising_Edge	16 bits GPIO Pins interrupt

7.4 Drivers Usage

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

Table 7-3 Usage of GPIO Drivers

Name	Description
GPIO_Init	Initialize GPIOx
GPIO_SetOutEnable	Set GPIOx output enable
GPIO_ClrOutEnable	Clear GPIOx output enable
GPIO_GetOutEnable	Return GPIOx output enable
GPIO_SetBit	GPIO output one
GPIO_ResetBit	GPIO output zero
GPIO_WriteBits	GPIO output
GPIO_ReadBits	GPIO input
GPIO_SetAltFunc	Set GPIOx alternate function enable
GPIO_ClrAltFunc	Clear GPIOx alternate function enable
GPIO_GetAltFunc	Return GPIOx alternate function enable
GPIO_IntClear	Clear GPIOx interrupt request
GPIO_GetIntStatus	Return GPIOx interrupt status
GPIO_SetIntEnable	Set GPIOx interrupt enable Return GPIOx interrupt status
GPIO_ClrIntEnable	Clear GPIOx interrupt enable Return GPIOx interrupt enable
GPIO_SetIntHighLevel	Setup GPIOx interrupt as high level
GPIO_SetIntRisingEdge	Setup GPIOx interrupt as rising edge
GPIO_SetIntLowLevel	Setup GPIOx interrupt as low level
GPIO_SetIntFallingEdge	Setup GPIOx interrupt as falling edge
GPIO_MaskedWrite	Setup GPIOx output value using masked access

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 8-1 Definition of I²C Master Registers

Register Name	Address Offset	Type	Width	Initial Value	Description
PRER	0x00	RW	32	0x0000FFFF	Clock prescale register [31:15] Reserved [15:0] Prescale value = sys_clk/(5*SCL)-1
CTR	0x04	RW	32	0x00000000	[31:8] Reserved [7] Enable I2C function [6] Enable I2C interrupt [5:0] Reserved
TXR	0x08	WO	32	0x00000000	[31:8] Reserved [7:1] Next transmission data [0] Data direction
RXR	0x0C	RO	32	0x00000000	[31:8] Reserved [7:0] Last received data
CR	0x010	WO	32	0x00000000	[31:8] Reserved [7] STA, Start transmission

Register Name	Address Offset	Type	Width	Initial Value	Description
					status [6] STO, Over transmission status [5] RD, Read enable, read data from slave [4] WR, Write enable, write data to slave [3] Acknowledge [2:1] Reserved [0] Interrupt acknowledge
SR	0x14	RO	32	0x00000000	[31:8] Reserved [7] Receive acknowledge signal from slave [6] I2C busy status [5] Arbitration loss [4:2] Reserved [1] Data transmission status flag [0] Interrupt flag

8.3 Drivers Usage

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

Table 8-2 Usage of I²C Master Drivers

Name	Description
I2C_Init	I²C Initialization
I2C_SendByte	Send a byte to I²C bus
I2C_SendBytes	Send multiple bytes to I²C bus
I2C_SendData	Send multiple bytes to I²C bus once time
I2C_ReceiveByte	Read a byte from I²C bus
I2C_ReadBytes	Read multiple bytes from I²C bus
I2C_ReceiveData	Read multiple bytes from I²C bus once time
I2C_Rate_Set	Set I²C traffic rate
I2C_Enable	Enable I²C bus
I2C_UnEnable	Disable I²C bus
I2C_InterruptOpen	Open I²C interrupt
I2C_InterruptClose	Close I²C interrupt

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

Register Name	Address Offset	Type	Width	Initial Value	Description
RDATA	0x00	RO	32	0x00000000	Read data register [31:8] Reserved [7:0] Read data
WDATA	0x04	WO	32	0x00000000	Write data register [31:8] Reserved [7:0] Write data
STATUS	0x08	RW	32	0x00000000	[31:8] Reserved [7] Overflow error status [6] Receive ready status [5] Transmit ready status [4] Be transmitting [3] Transmit overrun error status [2] Receive overrun error status [1:0] Reserved
SSMASK	0x0C	RW	32	0x00000000	[31:1] Reserved [0] Select and enable slave

Register Name	Address Offset	Type	Width	Initial Value	Description
CTRL	0x10	RW	32	0x00000000	[31:5] Reserved [4:3] Clock selected, CLK_I / 2/4/6/8 [2] Clock polarity [1] Clock polarity [0] Direction, 1 is MSB first

9.3 Initialization

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

Table 9-2 SPI Master Initialization

Name	Type	Value	Description
DIRECTION	FunctionalState	ENABLE/DISABLE	MSB/LSB first transmission
PHASE	FunctionalState	ENABLE/DISABLE	Posedge/Negedge transmit data
POLARITY	FunctionalState	ENABLE/DISABLE	Initialize ploarity to one/zero
CLKSEL	uint32_t	CLKSEL_CLK_DIV_2 CLKSEL_CLK_DIV_4 CLKSEL_CLK_DIV_6 CLKSEL_CLK_DIV_8	Select clock divided 2/4/6/8

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

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

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.

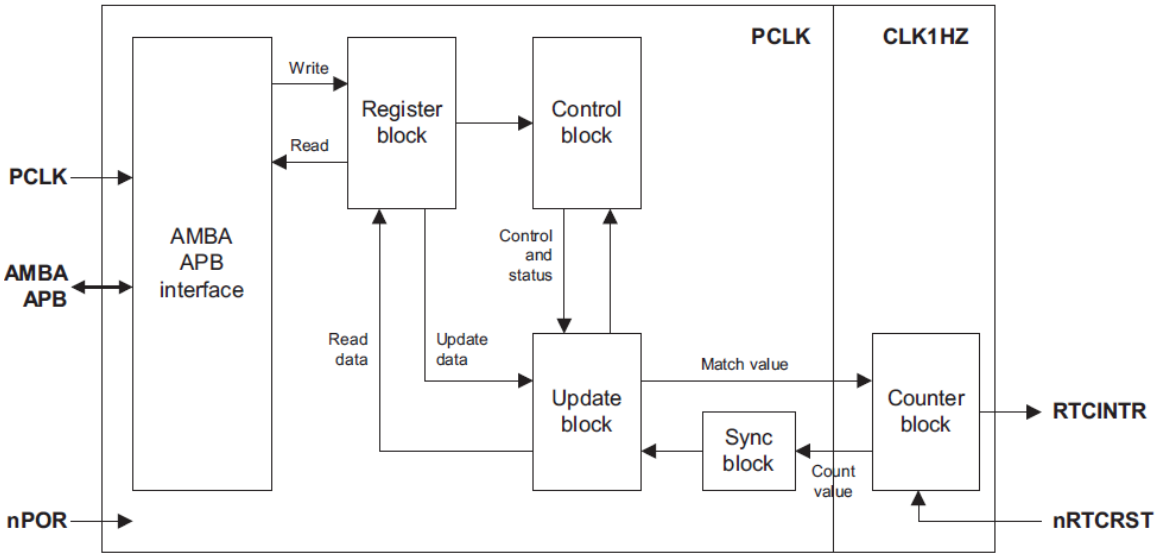
Realize RTC load or update via the write load register RTC_LOAD_VALUE

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.

Figure 10-1 RTC Block



10.2 Registers

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

Table 10-1 Definition of I²C Registers

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

Register Name	Address Offset	Type	Width	Initial Value	Description
					interrupt status
RTC_MIS	0x018	RO	32	0x00000000	Masked interrupt status register Get current masked interrupt status [31:1] Reserved [0] Current masked interrupt status
RTC_INTR_CLEAR	0x01C	WO	32	0x00000000	Interrupt clear register Clear current interrupt [31:1] Reserved [0] Clear current interrupt

10.3 Drivers Usage

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

Table 10-2 Usage of RTC Drivers

Name	Description
RTC_init	Initialize RTC
Get_Current_Value	Get RTC current value of data register
Set_Match_Value	Set RTC match value of match register
Get_Match_Value	Get RTC match value of match register
Set_Load_Value	Set RTC load value of load register
Get_Load_Value	Get RTC load value of load register
Start_RTC	Start RTC counter
Close_RTC	Cloase RTC counter
RTC_Inter_Mask_Set	Set RTC interrupt mask
Get_RTC_Control_value	Get value of control register
RTC_Inter_Mask_Clr	Clear RTC interrupt mask
Get_RTC_Inter_Mask_value	Get RTC interrupt mask
Clear_RTC_interrupt	Clear RTC interrupt

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, a10K 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 11-1 Definition of TRNG Registers

Register Name	Address Offset	Type	Width	Initial Value	Description
RESERVE1	0x000-0x0FC	-	-	-	Reserved
RNG_IMR	0x100	RW	32	0x0000000F	Interrupt mask register [31:4] Reserved [3] Mask the Von Neumann error [2] Mask the CRNGT error [1] Mask the Autocorrelation error [0] Mask when the TRNG has collected 192 bits
RNG_ISR	0x104	RO	32	0x00000000	Interrupt status register [31:4] Reserved [3] A Von Neumann error [2] A Continuos Random Number Generation Testing (CRNGT) error [1] The Autocorrelation test failed four times in a row

Register Name	Address Offset	Type	Width	Initial Value	Description
					[0] Set to 1 When 192 bits have been collected, and EHR_DATA[0-5] registers are ready to be read
RNG_ICR	0x108	WO	32	0x00000000	Interrupt clear register [31:4] Reserved [3] Clear a Von Neumann error [2] Clear a CRNGT error [1] Software cannot clear this bit, only a TRNG reset can clear this bit [0] Set to 1 after EHR_DATA[0-5] have been read
TRNG_CONFIG	0x10C	RW	32	0x00000000	Configuration register [31:2] Reserved [1:0] Selects the number of inverters: 00 = Selects the shortest inverter chain length 01 = Selects the short inverter chain length 10 = Selects the long inverter chain length 11 = Selects the longest inverter chain length
TRNG_VALID	0x110	RO	32	0x00000000	Valid register [31:1] Reserved [0] TRNG is complete, data can be read from EHR_DATA[0-5]
EHR_DATA0	0x114—x128	RO	32	0x00000000	Entropy holding register data register Return 32 bits from the 192-bit EHR DATA0 returns bits[31:0] DATA1 returns bits[63:32] DATA2 returns bits[95:64] DATA3 returns bits[127:96] DATA4 returns bits[159:128] DATA5 returns bits[191:160]
EHR_DATA1					
EHR_DATA2					
EHR_DATA3					
EHR_DATA4					
EHR_DATA5					

Register Name	Address Offset	Type	Width	Initial Value	Description
RND_SOURCE_ENABLE	0x12C	RW	32	0x00000000	Random source enable register [31:1] Reserved [0] 1 = enable entropy source; 0 = disable entropy source
SAMPLE_CNT1	0x130	RW	32	0x0000FFFF	Sample count register [31:0] Sets the number of rng_clk cycles
AUTOCORR_STATISTIC	0x134	RW	32	0x00000000	Autocorrelation register [31:22] Reserved [21:14] Count each time an autocorrelation test fails [13:0] Count each time an autocorrelation test starts
TRNG_DEBUG_CONTROL	0x138	RO	32	0x00000000	Debug control register [31:4] Reserved [3] The autocorrelation test is bypassed [2] The CRNGT test is bypassed [1] The Von Neumann balancer is bypassed [0] Reserved
RESERVE2	0x13C	-	-	-	Reserved
TRNG_SW_RESET	0x140	WO	32	0x00000000	Reset register [31:1] Reserved [0] Writing 1 to this register causes an internal TRNG reset
RESERVE3	0x144-0x1B4	-	-	-	Reserved
TRNG_BUSY	0x1B8	RO	32	0x00000000	Busy register [31:1] Reserved [0] Reflects the status of rng_busy signal
RST_BIT_COUNT	0x1BC	WO	32	0x00000000	Reset bits counter register [31:1] Reserved [0] Write any value to this bit resets the bits counter and TRNG valid registers
RESERVE4	0x1C0-0x1DC	-	-	-	Reserved
RNG_BIST_CNTR0	0x1E0-0x1E8	RO	32	0x00000000	BIST counter registers Return the collected BIST results
RNG_BIST_CNT					

Register Name	Address Offset	Type	Width	Initial Value	Description
R1					[31:22] Reserved
RNG_BIST_CNT R2					[21:0] Returns the results of the TRNG BIST counter

11.3 Drivers Usage

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

Table 11-2 Usage of TRNG Driver

Name	Description
Init_TRNG	Initialized TRNG
Set_Interrupt_Mask	Set interrupt mask
Get_Int_State	Get interrupt status
Clear_Int	Clear interrupt
Set_Config	Set config register
Get_EHR_Data	Get Entropy holding data
Set_Random_Source_Enable	Set random source enable register
Clr_Random_Source_Enable	Clear random source enable register
Set_Sample_Count	Set sample count register
Trng_SW_Reset	Reset TRNG
Get_TRNG_State	Get TRNG state
Reset_Bit_Count	Reset bit count register
Get_BIT_Counter	Get bits count register
Set_Debug_Control	Set debug control register
Fail_Start_State_times	Get autocorrelation register
Clr_Fail_Start_State_register	Clear autocorrelation register

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 12-1 Definition of DualTimer Registers

Register Name	Address Offset	Type	Width	Initial Value	Description
TIMER1LOAD	0x00	RW	32	0x00000000	Timer1 load register [31:0] Timer1 load value
TIMER1VALUE	0x04	RO	32	0xFFFFFFFF	Timer1 current value register [31:0] Timer1 current value
TIMER1CONTROL	0x08	RW	32	0x00000020	Timer1 control register [31:8] Reserved [7] Timer enable [6] Timer mode [5] Interrupt enable [4] Reserved [3:2] Timer prescale 00 = clock is divided by 1 01 = clock is divided by 16 10 = clock is divided by 256 11 Undefined [1] Timer size 0 = 16-bit counter, default 1 = 32-bit counter [0] One-shot count 0 = wrapping mode, default 1 = one-shot mode

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

Register Name	Address Offset	Type	Width	Initial Value	Description
					[0] Enable interrupt status from the counter
TIMER2BGLOAD	0x18	RW	32	0x00000000	Timer2 background load register [31:0] The value used to reload the counter

12.3 Drivers Usage

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

Table 12-2 Usage of DualTimer Drivers

Name	Description
DUALTIMER1_Init	Initialized DualTimer1
DUALTIMER2_Init	Initialized DualTimer2
Clear_DULATIMER_interrupt	Clear DualTimer interrupt
Dtimer_MODE_function	Set timer mode of DualTimer1 or DualTimer2
Dtimer_PRE_function	Set timer prescale of DualTimer1 or DualTimer2
INIT_NUM_load_function	Set load value of DualTimer1 or DualTimer2
ENABLE_interrupt_Dtimer_function	Enable interrupt of DualTimer1 or DualTimer2
TIMER_SIZE_function	Set timer size of DualTimer1 or DualTimer2
ENABLE_Dtimer_function	Enable DualTimer1 or DualTime2
Get_DULATIMER_interrupt_num	Get timer ID of DualTimer1 or DualTimer2

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 13-1 Definition of SD-Card Registers

Register Name	Address Offset	Type	Width	Initial Value	Description
SPI_MASTER_VERSION	0x000	RW	8	0x00	SPI master version register [7:4] Major revision number [3:0] Minor revision number
SPI_MASTER_CONTROL	0x001	WO	8	0x00	SPI master control register [7:1] Reserved [0] Reset core logic and register 1 = Reset core logic and register, self clearing
TRANS_TYPE	0x002	RW	8	0x00	Transaction type register [7:2] Reserved [1:0] Set the transaction type 00 = Direct access 01 = Initialized SD 10 = Read SD block

Register Name	Address Offset	Type	Width	Initial Value	Description
					11 = Write SD block
TRANS_CTRL	0x003	WO	8	0x00	Transaction control register [7:1] Reserved [0] Start transaction 1 = Start transaction, self clearing
TRANS_STS	0x004	RO	8	-	Transaction status register [7:1] Reserved [0] Transaction busy 1 = Transaction busy
TRANS_ERROR	0x005	RO	8	-	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 [1:0] SD initialize error 00 = Initialize no error 01 = Initialize command 0 error 10 = Initialize command 1 error
DIRECT_ACCESS_DATA	0x006	RW	8	0x00 / -	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
SD_ADDR_7_0	0x007	RW	8	0x00	SD address[7:0] bits register [7:0] SD_ADDR[7:0]
SD_ADDR_15_8	0x008	RW	8	0x00	SD address[15:8] bits register [7:0] SD_ADDR[15:8]
SD_ADDR_23_16	0x009	RW	8	0x00	SD address[23:16] bits register [7:0] SD_ADDR[23:16]
SD_ADDR_31_24	0x00A	RW	8	0x00	SD address[31:24] bits register [7:0] SD_ADDR[31:24]
SPI_CLK_DEL	0x00B	RW	8	0x00	SPI clock control register [7:0] Control the frequency of the SPI_CLK after SD initialization is completed

Register Name	Address Offset	Type	Width	Initial Value	Description
RESERVED0	0x00C-0x00F	-	-	-	Reserved
RX_FIFO_DATA	0x010	RW	8	-	SPI block reading data register [7:0] SD/MMC block read data, fifo size matches the SD/MMC block size of 512 bytes
RESERVED1	0x011	-	-	-	Reserved
RX_FIFO_DATA_COUNT_MSB	0x012	RO	8	-	MSB byte of reading data count register [7:0] MSByte of FIFO_DATA_COUNT, indicates the number of data entries within the fifo
RX_FIFO_DATA_COUNT_LSB	0x013	RO	8	-	LSB byte of reading data count register [7:0] LSByte of FIFO_DATA_COUNT, indicates the number of data entries within the fifo
RX_FIFO_CONTROL	0x014	WO	8	0x00	SD block reading data control register [7:1] Reserved [0] Force fifo empty 1 = Force fifo empty, delete all the data samples within the fifo, self clearing
RESERVED2	0x015-0x019	-	-	-	Reserved
TX_FIFO_DATA	0x020	WO	8	-	SD block writing data register [7:0] SD/MMC block write data, fifo size matches the SD/MMC block size of 512 bytes
RESERVED3	0x021-0x023	-	-	-	Reserved
TX_FIFO_CONTROL	0x024	WO	8	0x00	SD block writing data control register [7:1] Reserved [0] Force fifo empty 1 = Force fifo empty, delete all the data samples within the fifo, self clearing

13.3 Drivers Usage

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

Table 13-2 Usage of SD-Card Drivers

Name	Description
SD_Init	SD hardware initialization
SD_BlockWrite	SD block write data, block size is 512 bytes.
SD_BlockRead	SD block read data, block size is 512 bytes.

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 14-1 Definition of CAN Registers

Register Name	Address Offset	Type	Width	Initial Value	Description
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 [13:8] phseg2_nom,

Register Name	Address Offset	Type	Width	Initial Value	Description
					PHASE_SEG2 window's width [5:0] phseg1_nom, PROP_SEG+PHASE_SEG1 window's width
BTD	0x0010	RW	32	0x00000000	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
RSVD0	0x001C	-	-	-	Reserved
IS	0x0020	RO	32	0x00000000	Interrupt status register [31] Bus off status [27] TX message successfully [26] TX message retry [25] TX message failed [23] TX high-priority message successfully [22] TX high-priority message retry [21] TX high-priority message failed [8] Error status [5] TX high-priority fifo overflow [4] TX fifo overflow [1] RX fifo overflow [0] RX fifo valid
IE	0x0024	RW	32	0x00000000	Interrupt enable register [31] Enable us off [27] Enable TX message successfully [26] Enable TX message retry [25] Enable TX message failed [23] Enable TX high-priority message successfully [22] Enabe TX high-priority message retry [21] Enable TX high-priority message failed [8] Enable Error status [5] Enabe TX high-priority fifo overflow [4] Enable TX fifo overflow [1] Enable RX fifo overflow [0] Enable RX fifo valid
IC	0x0028	WO	32	0x00000000	Interrupt clear register [31] Clear bus off status [27] Clear TX message

Register Name	Address Offset	Type	Width	Initial Value	Description
					successfully status [26] Clear TX message retry status [25] Clear TX message failed status [23] Clear TX high-priority message successfully status [22] Clear TX high-priority message retry status [21] Clear TX high-priority message failed status [8] Clear Error status status [5] Clear TX high-priority fifo overflow status [4] Clear TX fifo overflow status [1] Clear RX fifo overflow status [0] Clear RX fifo valid status
RSVD1	0x002C	-	-	-	Reserved
CFG	0x0030	RW	32	0x00000000	Configuration register [4] Configure disprotexceponres 1 = 'res' is FORM-ERROR 0 = 'res' is exception [0] Configure isofd 1 = ISO FD mode 0 = non ISO FD mode
RSVD2	0x0034-0x003C	-	-	-	Reserved
RXBCFG	0x0040	RW	32	0x00000000	RX buffer/fifo configuration register [31:16] RX buffer's ending offset [15:0] RX buffer's start offset
TXBCFG	0x0044	RW	32	0x00000000	TX buffer/fifo configuration register [31:16] TX buffer's ending offset [15:0] TX buffer's start offset
TXHBCFG	0x0048	RW	32	0x00000000	TX high-priority/fifo configuration register [31:16] TX high-priority buffer's ending offset [15:0] TX high-priority buffer's start offset
RSVD3	0x004C	-	-	-	Reserved
TXBRETRY	0x0050	RW	32	0x00000000	TX buffer retry counter
TXHBRETRY	0x0054	RW	32	0x00000000	TX high-priority buffer retry counter
TXMSGSTS	0x0058	RO	32	0x00000000	Transmit message status register [31:30] TX message status 00 = successfully

Register Name	Address Offset	Type	Width	Initial Value	Description
					10 = retry 11 = failed [28:0] Message ID
TXHMSGSTS	0x005C	RO	32	0x00000000	Transmit high-priority message status register [31:30] TX high-priority message status 00 = successfully 10 = retry 11 = failed [28:0] Message ID
ERRSTS	0x0060	RW	32	0x00000000	Error status register [4] CRC error [3] ACK error [2] FORM error [1] BIT error [0] STUFF error
ERRCNTR	0x0064	RO	32	0x00000000	Error counter register [24:16] TX error counter [8:0] RX error counter
RSVD4	0x0068-0x00fc	-	-	-	Reserved
AF	0x0100-0x100+(4*N)	RW	32	0x00000000	Receive acceptance filter register [31] Enable 1 = valid 0 = invalid [30] IDE 1 = extended frame 0 = normal frame [29] Extended data length 1 = match FD frame 0 = match normal frame [28:18] Basic ID [17:0] ID Extension
AFM	0x0140-0x140+(4*N)	RW	32	0x00000000	Receive acceptance filter mask register [28:18] Basic ID mask [17:0] ID Extension mask
RSVD5	0x0180-0x01FC	-	-	-	Reserved
RXB	0x0200	RO	32	0x00000000	Receive buffer/fifo window register
TXB	0x0204	WO	32	0x00000000	Transmit buffer/fifo window register
TXHB	0x0208	WO	32	0x00000000	Transmit high-priority buffer/fifo window register
TXBSTS	0x020C	RO	32	0x00000000	Transmit buffer/fifo status [31] txbwerr

Register Name	Address Offset	Type	Width	Initial Value	Description
					[15:0] txbspace
TXHBSTS	0x0210	RO	32	0x00000000	Transmit high-priority buffer/fifo status [31] txhbwerr [15:0] txhbspace
RXBSTS	0x0214	RO	32	0x00000000	Receive buffer/fifo status [15:0] rxbdepth

14.3 Driver Usage

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

Table 14-2 Usage of CAN Drivers

Name	Description
can_srst	Start hard reset
can_set_cmd	Enable working mode
can_set_brp	Set baud rate prescalar
can_set_btn_phseg1_nom	Set PROP_SEG+PHASE_SEG1 window's width
can_set_btn_phseg2_nom	Set PHASE_SEG2 window's width
can_set_btn_sjw_nom	Set sjw_nom
can_set_btn	Set BTN register
can_read_btn_phseg1_nom	Get PROP_SEG+PHASE_SEG1 window's width
can_read_btn_phseg2_nom	Get PHASE_SEG2 window's width
can_read_btn_sjw_nom	Get sjw_nom
can_set_btd_phseg1_d	Set PROP_SEG+PHASE_SEG1 window's depth
can_set_btd_phseg2_d	Set PHASE_SEG2 window's depth
can_set_btd_sjw_d	Set sjw_d
can_set_btd	Set BTD register
can_read_btd_phseg1_d	Get PROP_SEG+PHASE_SEG1 window's depth
can_read_btd_phseg2_d	Get PHASE_SEG2 window's depth
can_read_btd_sjw_d	Get sjw_d
can_read_is_bit	Get IS register bits function
can_set_ie_bit	Set IE register bits function
can_clear_ie_bit	Clear IE register bits function
can_read_ie_bit	Get IE register bits function
can_set_ic_bit	Set IC register bits function
can_set_cfg_bit_as_one	Set CFG register bits function
can_set_cfg_bit_as_zero	Clear CFG register bits function
can_read_cfg_bit	Get CFG register bits function
can_set_rxbcfg_rxb_start	Set RX buffer start offset in RXBCFG
can_read_rxbcfg_rxb_start	Get RX buffer start offset in RXBCFG

Name	Description
can_set_rxbcfg_rxb_end	Set RX buffer ending offset in RXBCFG
can_read_rxbcfg_rxb_end	Get RX buffer ending offset in RXBCFG
set_rxbcfg	Set RX buffer start and ending offset
can_set_txbcfg_txb_start	Set TX buffer start offset in TXBCFG
can_read_txbcfg_txb_start	Get TX buffer start offset in TXBCFG
can_set_txbcfg_txb_end	Set TX buffer ending offset in TXBCFG
can_read_txbcfg_txb_end	Get TX buffer ending offset in TXBCFG
set_txbcfg	Set TX buffer start and ending offset
can_set_txhbcfg_txhb_start	Set TX high-priority buffer start offset in TXHBCFG
can_read_txhbcfg_txhb_start	Get TX high-priority buffer start offset in TXHBCFG
can_set_txhbcfg_txhb_end	Set TX high-priority buffer ending offset in TXHBCFG
can_read_txhbcfg_txhb_end	Get TX high-priority buffer ending offset in TXHBCFG
set_txhbcfg	Set TX high-priority buffer start and ending offset
can_set_txbretry	Set TX buffer retry
can_read_txbretry	Get TX buffer retry
can_set_txhbretry	Set TX high-priority buffer retry counter
can_read_txhbretry	Get TX high-priority buffer retry counter
can_read_txmsgsts	Get TX message status
can_read_txmsgid	Get TX message ID
can_read_txhmsgsts	Get TX high-priority message status
can_read_txhmsgid	Get TX high-priority message ID
can_read_errsts	Get error status
can_read_errcntr_rec	Get RX error counter
can_read_errcntr_tec	Get TX error counter
can_set_af_bit_as_one	Set AF bits function
can_set_af_bit_as_zero	Clear AF bits function
can_read_af_bit	Get AF bits function
can_set_af_ie	Set AF ID Extension
can_read_af_ie	Get AF ID Extension
can_set_af_bid	Set AF basic ID
can_read_af_bid	Get AF basic ID
can_set_afm_iem	Set AFM ID Extension mask
can_read_afm_iem	Get AFM ID Extension mask
can_set_afm_bidm	Set AFM basic ID mask
can_read_afm_bid	Get AFM basic ID mask
can_read_rxb	Get RXB register
can_set_txb	Set TXB register
can_set_txhb	Set TXHB register

Name	Description
can_read_txbsts_tdbspace	Get txbspace
can_read_txbsts_txbwerr	Get txbwerr
can_read_txhbsts_tdbspace	Get txhbspace
can_read_txhbsts_txbwerr	Get txhbwerr
can_read_rxbsts	Get rxbdepth

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 15-1 Definition of Ethernet Registers

Name	Address Offset	Type	Width	Initial Value	Description
ETH_TX_DATA	0x000-0x5FF	WO	32	0x00000000	Transmit data registers
ETH_RX_DATA	0x000-0x5FFF	RO	32	0x00000000	Receive data registers
ETH_TX_LENGTH	0x600	RW	32	0x00000000	Transmit data length [31:11] Reserved [10:0] TX data length
ETH_TX_EN	0x604	RW	32	0x00000000	Transmit enable [31:1] Reserved [0] Enable TX
ETH_TX_FAIL	0x608	RW	32	0x00000000	Transmit failed status [31:3] Reserved [2] TX late [1] TX excessive [0] TX failed
ETH_TX_IS	0x60C	RO	32	0x00000000	Transmit interrupt status [31:1] Reserved [0] TX interrupt status
ETH_TX_IC	0x610	WO	32	0x00000000	Transmit interrupt clear [31:1] Reserved [0] Clear TX interrupt
ETH_TX_IE	0x614	RW	32	0x00000000	Transmit interrupt enable [31:1] Reserved [0] Enable TX interrupt
RESERVED_1	0x618-0x67F	–	–	–	Reserved
ETH_RX_LEGHT	0x680	RO	32	0x00000000	Receive data length
ETH_RX_IS	0x684	RO	32	0x00000000	Receive interrupt status [31:1] Reserved [0] RX interrupt status
ETH_RX_IE	0x688	RW	32	0x00000000	Receive interrupt enable [31:1] Reserved [0] Enable RX interrupt
ETH_RX_IC	0x68C	WO	32	0x00000000	Receive interrupt clear [31:1] Reserved [0] Clear RX interrupt
RESERVED_2	0x690-0x6FFF	–	–	–	Reserved
MIIM_OP_MODE	0x700	RW	32	0x00000000	MIIM operation mode [31:1] Reserved [0] MIIM operation mode

Name	Address Offset	Type	Width	Initial Value	Description
MIIM_PHY_ADDR	0x704	RW	32	0x00000000	MIIM PHY address [31:5] Reserved [4:0] MIIM PHY address
MIIM_REG_ADDR	0x708	RW	32	0x00000000	MIIM reg address [31:5] Reserved [4:0] MIIM reg address
MIIM_WR_DATA	0x70C	RW	32	0x00000000	MIIM write data [31:16] Reserved [15:0] MIIM write data
MIIM_RD_DATA	0x710	RO	32	0x00000000	MIIM read data [31:16] Reserved [15:0] MIIM read data
MIIM_IS	0x714	RO	32	0x00000000	MIIM interrupt status [31:2] Reserved [1] MIIM operation end [0] MIIM read data valid
MIIM_IE	0x718	RW	32	0x00000000	MIIM interrupt enable [31:2] Reserved [1] MIIM operation end [0] MIIM read data valid
MIIM_IC	0x71C	WO	32	0x00000000	MIIM interrupt clear [31:2] Reserved [1] MIIM operation end [0] MIIM read data valid
MIIM_OP_EN	0x720	RW	32	0x00000000	MIIM operation enable [31:1] Reserved [0] Enable MIIM operation
ETH_MODE	0x724	RW	32	0x00000000	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

15.3 Drivers Usage

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

Table 15-2 Usage of Ethernet Drivers

Name	Description
eth_init	Initialize Ethernet
tx_int_event	TX interrupt
rx_int_event	RX interrupt
eth_tx	Ethernet TX
eth_set_mode	Set Ethernet duplex mode and speed
miim_wr_int_event	MIIM interface transmits interrupt
miim_rd_int_event	MIIM interface receives interrupt
miim_write	MIIM interface transmits data
miim_receive	MIIM interface receives data

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 16-1 Definition of DDR3 Registers

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

16.3 Drivers Usage

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

Table 16-2 Usage of DDR3 Drivers

Name	Description
DDR3_Init	Initialize DDR3
DDR3_Read	Read data from DDR3
DDR3_Write	Write data into DDR3

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 W25Q64 BV 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

Name	Address Offset	Type	Width	Initial Value	Description
IDREV	0x00	RO	32	0x02002000	ID and revision register [31:8] ID number [7:4] Major revision number [3:0] Minor revision number
RESERVED0[3]	0x04-0x0C	-	-	-	Reserved
TRANSFMT	0x10	RW	32	0x00020780	SPI transfer format register [31:18] Reserved [17:16] Address length in bytes 00 = 1 byte 01 = 2 bytes 10 = 3 bytes 11 = 4 bytes [15:13] Reserved [12:8] Data length [7] Enable data merge mode [6:5] Reserved [4] Bi-directional MOSI in single mode 0 = MOSI is uni-directional signal

Name	Address Offset	Type	Width	Initial Value	Description
					1 = MOSI is bi-directional signal [3] Transfer data with the lease 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
DIRECTIO	0x14	RW	32	0x0	SPI direct IO control register [31:25] Reserved [24] Enable direct IO 0 = Disable 1 = Enable [23:22] Reserved [21] Output enable for SPI-Flash hold signal [20] Output enable for SPI-Flash write protect signal [19] Output enable for the SPI MISO signal [18] Output enable for the SPI MOSI signal [17] Output enable for SPI SCLK signal [16] Output enable for SPI CS signal [15:14] Reserved [13] Output value for SPI-Flash hold signal [12] Output value for SPI-Flash write protect signal [11] Output value for SPI MISO signal [10] Output value for SPI MOSI signal [9] Output value for SPI SCLK signal [8] Output value for SPI CS signal [7:6] Reserved

Name	Address Offset	Type	Width	Initial Value	Description
					[5] Status of SPI-Flash hold signal [4] Status of SPI-Flash write protect signal [3] Status of SPI MISO signal [2] Status of SPI MOSI signal [1] Status of SPI SCLK signal [0] Status of SPI CS signal
RESERVED1[2]	0x18-0x1C	-	-	-	Reserved
TRANSCTRL	0x20	RW	32	0x0	SPI transfer control register [31] Reserved [30] SPI command phase enable 0 = Disable the command phase 1 = Enable the command phase (Master mode only) [29] SPI address phase enable 0 = Disable the address phase 1 = Enable the address phase (Master mode only) [28] SPI address phase format 0 = Address phase is single mode 1 = The format of the address phase is the same as the DualQuad data phase (Master mode only) [27:24] Transfer mode 0000 = Write and read at the same time 0001 = Write only 0010 = Read only 0011 = Write, Read 0100 = Read, Write 0101 = Write, Dummy, Read 0110 = Read, Dummy, Write 0111 = None data 1000 = Dummy, Write 1001 = Dummy, Read 1010~1111 = Reserved [23:22] SPI data phase format 00 = Single mode 01 = Dual I/O mode 10 = Quad I/O mode 11 = Reserved [21] Append and one-byte special token following the address phase for SPI read transfers [20:12] Transfer count for write data [11] The value of the one-byte special token following the

Name	Address Offset	Type	Width	Initial Value	Description
					address phase for SPI read transfers 0 = token value is 0x00 1 = token value is 0x69 [10:9] Dummy data count [8:0] Transfer count for read data
CMD	0x24	RW	32	0x0	SPI command register [31:8] Reserved [7:0] SPI command
ADDR	0x28	RW	32	0x0	SPI address register [31:0] SPI address (Master mode only)
DATA	0x2C	RW	32	0x0	SPI data register [31:0] Data to transmit or the received data
CTRL	0x30	RW	32	0x0	SPI controller register [31:21] Reserved [20:16] Transmit FIFO threshold [15:13] Reserved [12:8] Receive FIFO threshold [7:5] Reserved [4] TX DMA enable [3] RX DMA enable [2] Transmit FIFO reset [1] Receive FIFO reset [0] SPI reset
STATUS	0x34	RO	32	0x0	SPI status register [31:24] Reserved [23] Transmit FIFO full flag [22] Transmit FIFO empty flag [21] Reserved [20:16] Number of valid entries int the transmit FIFO [15] Receive FIFO full flag [14] Receive FIFO empty flag [13] Reserved [12:8] Number of valid entries in the receive FIFO [7:1] Reserved [0] SPI register programming is in progress
INTREN	0x38	RW	32	0x0	SPI interrupt enable register [31:6] Reserved [5] Enable the slave command interrupt [4] Enable the end of SPI transfer interrupt [3] Enable the SPI transmit FIFO

Name	Address Offset	Type	Width	Initial Value	Description
					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)
INTRST	0x3C	WO	32	0x0	SPI interrupt status register [31:6] Reserved [5] Slave command interrupt (Slave mode only) [4] End of SPI transfer interrupt [3] TX FIFO threshold interrupt [2] RX FIFO threshold interrupt [1] TX FIFO underrun interrupt (Slave mode only) [0] RX FIFO overrun interrupt (Slave mode only)
TIMING	0x40	RW	32	0x0	SPI interface timing register [31:14] Reserved [13:12] The minimum time between the edges of SPI CS and the edges of SCLK [11:8] The minimum time the SPI CS should stay HIGH [7:0] The clock frequency ratio between the clock source and SPI interface SCLK
RESERVED2[3]	0x44-0x4c	-	-	-	Reserved
MEMCTRL	0x50	RW	32	0x0	SPI memory access control register [31:9] Reserved [8] This bit is set when “MEMCTRL” / ”TIMING” is written [7:4] Reserved [3:0] Selects the SPI command
RESERVED3[3]	0x54-0x5C	-	-	-	Reserved
SLVST	0x60	RW	32	0x0	SPI slave status register [31:19] Reserved [18] Data underrun occurs in the last transaction [17] Data overrun occurs in the last transaction [16] SPI is ready for data transaction [15:0] User defined status flags
SLVDATACNT	0x64	RO	32	0x0	SPI slave data count register

Name	Address Offset	Type	Width	Initial Value	Description
					[31:25] Reserved [24:16] Slave transmitted data count [15:9] Reserved [8:0] Slave received data count
RESERVED4[5]	0x68-0x78	-	-	-	Reserved
CONFIG	0x7C	RO	32	0x0	Configuration register [31:15] Reserved [14] Support for SPI slave mode [13] Reserved [12] Support for memory-mapped access through AHB bus [11] Support for direct SPI IO [10] Reserved [9] Support for Quad I/O SPI [9] Support for Dual I/O SPI [7:6] Reserved [5:4] Depth of TX FIFO 00 = 2 words 01 = 4 words 10 = 8 words 11 = 16 words [3:2] Reserved [1:0] Depth of RX FIFO 00 = 2 words 01 = 4 words 10 = 8 words 11 = 16 words

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

Name	Description
spi_flash_init	Initialize SPI-Flash
change_mode_spi_flash	Switch SPI-Flash mode between download and read, write, erase memory
spi_flash_read	Read data from SPI-Flash
spi_flash_write	Write data into SPI-Flash
spi_flash_write_read	Write data into SPI-Flash and read data from SPI-Flash once time
spi_flash_page_program	Write data into SPI-Flash with pages
spi_flash_sector_erase	Erase SPI-Flash with sector

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 18-1 Definition of PSRAM Registers

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

18.3 Drivers Usage

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

Table 18-2 Usage of PSRAM Drivers

Name	Description
PSRAM_Check_Init_Status	Check the status of PSRAM initialization
PSRAM_Mode_Set	Set the mode fo PSRAM write and read
PSRAM_Address_Set	Set the address of PSRAM and save data into this address
PSRAM_Read_Data_Buff	Read data from the buffer of PSRAM
PSRAM_Cmd_Enable	Enable the command of PSRAM
PSRAM_Read_Done_Flag	Get the flag of read PSRAM done
PSRAM_Clear_Read_Done_Flag	Clear the flag of read PSRAM done
PSRAM_Write_Data_Buff	Write data into the buffer of PSRAM
PSRAM_Cmd_Unable	Disable the command of PSRAM
PSRAM_Write_Data_Package	Write a package data into PSRAM
PSRAM_Read_Data_Package	Read a package data from PSRAM

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

19 Embedded Real-time Operating System

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.

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_ucos_iii

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\free_rtos
- Gowin_EMPU_M1\ref_design\MCU_RefDesign\GMD_RefDesign\cm1_freertos


GOWIN
PROGRAMMING FOR THE FUTURE