



Gowin_EMPU(GW1NS-4C) Software Programming **Reference Manual**

IPUG931-2.0E, 03/14/2024

Copyright © 2024 Guangdong Gowin Semiconductor Corporation. All Rights Reserved.

GOWIN is a trademark of Guangdong Gowin Semiconductor Corporation and is registered in China, the U.S. Patent and Trademark Office, and other countries. All other words and logos identified as trademarks or service marks are the property of their respective holders. No part of this document may be reproduced or transmitted in any form or by any denotes, electronic, mechanical, photocopying, recording or otherwise, without the prior written consent of GOWINSEMI.

Disclaimer

GOWINSEMI assumes no liability and provides no warranty (either expressed or implied) and is not responsible for any damage incurred to your hardware, software, data, or property resulting from usage of the materials or intellectual property except as outlined in the GOWINSEMI Terms and Conditions of Sale. GOWINSEMI may make changes to this document at any time without prior notice. Anyone relying on this documentation should contact GOWINSEMI for the current documentation and errata.

Revision History

Date	Version	Description
04/20/2020	1.0E	Initial version published.
02/08/2021	1.1E	● AHB PSRAM Memory Interface peripheral supported. ● AHB HyperRAM Memory Interface peripheral supported. ● APB SPI Nor Flash peripheral supported. ● GPIO supports multiple port types. ● I²C supports multiple port types. ● ARM Keil MDK as well as GOWIN MCU Designer versions upgraded.
06/21/2021	1.2E	Known issue of SPI full duplex read and write fixed.
12/16/2022	1.3E	● Known issue of SPI frequency division coefficient fixed. ● RT-Thread Nano RTOS reference design supported. ● Software development kit updated
03/14/2024	2.0E	Software programming library and software programming reference design updated.

Contents

Contents	i
List of Figures	v
List of Tables	vi
1 Software Programming Library	1
1.1 MCU Software Programming	1
1.2 Embedded RTOS Software Programming	3
2 Memory System	4
2.1 Standard Peripherals Memory Mapping	4
2.2 Core System Memory Mapping	5
3 Interrupt Handler	6
4 Universal Asynchronous Receiver/Transmitter	9
4.1 Features	9
4.2 Register Definition	11
4.3 Initialization Definition	12
4.4 Usage of Drivers	12
5 Timer	14
5.1 Features	14
5.2 Register Definition	15
5.3 Initialization Definition	15
5.4 Usage of Drivers	16
6 Watchdog	17
6.1 Features	17
6.2 Register Definition	18
6.3 Initialization Definition	19
6.4 Usage of Drivers	19

7	GPIO.....	21
7.1	Features	21
7.2	Register Definition	22
7.3	Initialization Definition.....	24
7.4	Usage of Drivers.....	25
8	Real-time Clock.....	27
8.1	Features	27
8.2	Register Definition	28
8.3	Usage of Drivers.....	29
9	SPI.....	31
9.1	Features	31
9.2	Register Definition	31
9.3	Initialization Definition.....	32
9.4	Usage of Drivers.....	32
10	System Controller.....	34
10.1	Register Definition	34
10.2	Usage of Drivers.....	34
11	I²C Master	36
11.1	Features.....	36
11.2	Register Definition	36
11.3	Usage of Drivers	37
12	SysTick	39
12.1	Features	39
12.2	Register Definition	39
12.3	Usage of Drivers.....	40
13	Memory Management.....	41
13.1	Features	41
13.2	Usage of Driver	41
14	SPI-Flash Memory.....	42
14.1	Features	42
14.2	Register Definition	42

14.3 Usage of Drivers.....	49
15 PSRAM Memory Interface	50
15.1 Features	50
15.2 Register Definition	51
15.3 Usage of Drivers.....	52
16 HyperRAM Memory Interface.....	53
16.1 Features	53
16.2 Register Definition	54
16.3 Usage of Drivers.....	55
17 Real-time Operating System	56
17.1 uC/OS-III	56
17.1.1 Features	56
17.1.2 Version	56
17.1.3 Configuration	56
17.2 FreeRTOS	57
17.2.1 Features	57
17.2.2 Version	57
17.2.3 Configuration	57
17.3 RT-Thread Nano Version	57
17.3.1 Features	57
17.3.2 Version	58
17.3.3 Configuration	58
18 Application Programs	59
18.1 UART	59
18.2 Timer.....	59
18.3 Watch Dog.....	60
18.4 GPIO.....	60
18.5 I ² C Master.....	60
18.6 SPI Master.....	60
18.7 RTC	61
18.8 SPI-Flash Memory.....	61
18.9 PSRAM Memory.....	61
18.10 HyperRAM Memory.....	61

18.11 Interrupt	62
18.12 DMM	62
18.13 AHB Master	62
18.14 APB Master	62
18.15 uC/OS-III	62
18.16 FreeRTOS	63
18.17 RT-Thread Nano Version	63

List of Figures

Figure 4-1 UART Block Diagram	10
Figure 5-1 TIMER Block Diagram	14
Figure 6-1 WatchDog Operating Flow	17
Figure 7-1 GPIO Block Diagram	21
Figure 8-1 RTC Block Diagram	28

List of Tables

Table 1-1 MCU Software Programming	1
Table 2-1 Definition of Standard Peripherals Memory Mapping	4
Table 2-2 Definition of Core System Memory Mapping	5
Table 3-1 Definition of Interrupt Controller	6
Table 4-1 Definition of UART Registers	11
Table 4-2 Definition of UART Initialization.....	12
Table 4-3 Usage of UART Drivers.....	12
Table 5-1 Definition of TIMER Registers.....	15
Table 5-2 Definition of TIMER Initialization	15
Table 5-3 Usage of TIMER Drivers	16
Table 6-1 Definition of Watchdog Registers.....	18
Table 6-2 Definition of WatchDog Initialization.....	19
Table 6-3 Usage of WatchDog Drivers.....	19
Table 7-1 Definition of GPIO Registers	22
Table 7-2 Definition of GPIO Initialization	24
Table 7-3 Usage of GPIO Drivers	25
Table 8-1 Definition of RTC Registers.....	28
Table 8-2 Usage of RTC Drivers	29
Table 9-1 Definition of SPI Master Registers	31
Table 9-2 Definition of SPI Master Initialization	32
Table 9-3 Usage of SPI Master Drivers	32
Table 10-1 Definition of SYSCON Register	34
Table 10-2 Usage of SYSCON Drivers	34
Table 11-1 Definition of I ² C Master Registers	36
Table 11-2 Usage of I ² C Master Drivers.....	37
Table 12-1 Definition of SysTick Registers	39
Table 12-2 Usage of SysTick Drivers.....	40

Table 13-1 Usage of Memory Management Drivers	41
Table 14-1 Definition of SPI-Flash Memory Registers	42
Table 14-2 Usage of SPI-Flash Memory Drivers	49
Table 15-1 Definition of PSRAM Memory Interface Registers	51
Table 15-2 Usage of PSRAM Memory Interface Drivers	52
Table 16-1 Definition of HyperRAM Memory Interface Registers	54
Table 16-2 Usage of HyperRAM Memory Interface Drivers.....	55

1 Software Programming Library

Gowin_EMPU(GW1NS-4C) supports software programming library: ...\library.

Gowin_EMPU(GW1NS-4C) supports two software programming methods:

- MCU software programming
- Embedded RTOS software programming

1.1 MCU Software Programming

Gowin_EMPU(GW1NS-4C) software programming library supports MCU software programming, as shown in Table 1-1.

Table 1-1 MCU Software Programming

File	Description
System Definition	
startup_gw1ns4c.s	Cortex-M3 core bootloader
core_cm3.c core_cm3.h	Cortex-M3 core register definition
gw1ns4c.h	Definition of interrupt vector table, register, and address mapping
system_gw1ns4c.c	System initialization and system clock definition
Linker Definition	
gw1ns4c_flash.ld	GMD IDE Flash linker
Peripheral Driver Definition	
gw1ns4c_gpio.c gw1ns4c_gpio.h	GPIO driver function definition
gw1ns4c_uart.c	UART driver function definition

File	Description
gw1ns4c_uart.h	
gw1ns4c_timer.c gw1ns4c_timer.h	Timer driver function definition
gw1ns4c_wdog.c gw1ns4c_wdog.h	Watchdog driver function definition
gw1ns4c_spi.c gw1ns4c_spi.h	SPI Master driver function definition
gw1ns4c_i2c.c gw1ns4c_i2c.h	I ² C Master driver function definition
gw1ns4c_rtc.c gw1ns4c_rtc.h	RTC driver function definition
gw1ns4c_misc.c gw1ns4c_misc.h	Interrupt priority and SysTick definition
gw1ns4c_syscon.c gw1ns4c_syscon.h	System control driver function definition
Middleware Definition	
malloc.c malloc.h	Dynamic memory management malloc and free function redefinition
retarget.c uart.c uart.h	UART0 printf function redefinition UART initialization definition
delay.c delay.h	Delay function definition
gpio.c gpio.h	GPIO initialization definition
psram.c psram.h	PSRAM Memory Interface drive function definition
hyper_ram.c hyper_ram.h	HyperRAM Memory Interface drive function definition
spi_flash.c spi_flash.h	SPI-Flash drive function definition

1.2 Embedded RTOS Software Programming

Gowin_EMPU(GW1NS-4C) supports the following three RTOS software programming:

- uC/OS-III
- FreeRTOS
- RT-Thread Nano version

2 Memory System

2.1 Standard Peripherals Memory Mapping

The standard peripherals memory mapping addresses for Gowin_EMPU(GW1NS-4C) are as shown in Table 2-1.

Table 2-1 Definition of Standard Peripherals Memory Mapping

Standard Peripheral	Type	Address Mapping	Description
FLASH	–	0x00000000	32KB
SRAM	–	0x20000000	2KB, 4KB, 8KB or 16KB
TIMER0	TIMER_TypeDef	0x40000000	Timer 0
TIMER1	TIMER_TypeDef	0x40001000	Timer 1
UART0	UART_TypeDef	0x40004000	UART0
UART1	UART_TypeDef	0x40005000	UART1
RTC	RTC_TypeDef	0x40006000	Real-time clock
WatchDog	WDog_TypeDef	0x40008000	Watchdog
GPIO0	GPIO_TypeDef	0x40010000	GPIO port
SYSCON	SYSCON_TypeDef	0x4001F000	System Control
I2C	I2C_TypeDef	0x40002000	I2C
SPI	SPI_TypeDef	0x40002200	SPI
APB2 Master 1	–	0x40002400	APB2 Master [1]
APB2 Master 2	–	0x40002500	APB2 Master [2]
APB2 Master 3	–	0x40002600	APB2 Master [3]
APB2 Master 4	–	0x40002700	APB2 Master [4]
APB2 Master 5	–	0x40002800	APB2 Master [5]
APB2 Master 6	–	0x40002900	APB2 Master [6]

Standard Peripheral	Type	Address Mapping	Description
APB2 Master 7	–	0x40002A00	APB2 Master [7]
APB2 Master 8	–	0x40002B00	APB2 Master [8]
APB2 Master 9	–	0x40002C00	APB2 Master [9]
APB2 Master 10	–	0x40002D00	APB2 Master [10]
APB2 Master 11	–	0x40002E00	APB2 Master [11]
APB2 Master 12	–	0x40002F00	APB2 Master [12]
AHB2 Master	–	0xA0000000	AHB2 Master

2.2 Core System Memory Mapping

Gowin_EMPU(GW1NS-4C) core system memory mapping definitions are as shown in Table 2-2.

Table 2-2 Definition of Core System Memory Mapping

System Control	Type	Address Mapping	Description
ITM	ITM_Type	0xE0000000	ITM configuration structure
DWT	DWT_Type	0xE0001000	DWT configuration structure
CoreDebug	CoreDebug_Type	0xE000EDF0	Core Debug configuration structure
ETM	ETM_Type	0xE0041000	ETM configuration structure
SysTick	SysTick_Type	0xE000E010	SysTick configuration structure
NVIC	NVIC_BASE	0xE000E100	NVIC configuration structure
SCnSCB	SCnSCB_Type	0xE000E000	System control Register not in SCB
SCB	SCB_Type	0xE000ED00	SCB configuration structure
TPIU	TPIU_Type	0xE0040000	TPIU configuration structure

3 Interrupt Handler

The nested vector interrupt controller (NVIC) includes the following features:

- Supports up to 32 low delay interrupts
- Provides 6 interrupt handling signals to users (USER_INT0~5)
- Supports programmable interrupt priorities of 0 -7
- Low delay interrupts and exception handling
- Interrupt signal edge or pulse detection
- Interrupt priority adjustment

Gowin_EMPU(GW1NS-4C) interrupt controller definitions are 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	MemManage_Handler	-12	MPU Fault Handler
0x00000014	BusFault_Handler	-11	Bus Fault Handler
0x00000018	UsageFault_Handler	-10	Usage Fault Handler
0x0000001C	0	–	Reserved
0x00000020	0	–	Reserved
0x00000024	0	–	Reserved
0x00000028	0	–	Reserved
0x0000002C	SVC_Handler	-5	SVCall Handler
0x00000030	DebugMon_Handler	-4	Debug Monitor Handler

Address	Interrupt	Number	Description
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	USER_INT0_Handler	1	16+ 1: USER_INT0
0x00000048	UART1_Handler	2	16+ 2: UART 1 RX and TX Handler
0x0000004C	USER_INT1_Handler	3	16+ 3: USER_INT1
0x00000050	USER_INT2_Handler	4	16+ 4: USER_INT2
0x00000054	RTC_Handler	5	16+ 5: RTC Handler
0x00000058	PORT0_COMB_Handler	6	16+ 6: GPIO Port 0 Combined Handler
0x0000005C	USER_INT3_Handler	7	16+ 7: USER_INT3
0x00000060	TIMER0_Handler	8	16+ 8: TIMER 0 Handler
0x00000064	TIMER1_Handler	9	16+ 9: TIMER 1 Handler
0x00000068	0	–	16+10: Reserved
0x0000006C	I2C_Handler	11	16+11: I2C Handler
0x00000070	UARTOVF_Handler	12	16+12: UART 0,1 Overflow Handler
0x00000074	USER_INT4_Handler	13	16+13: USER_INT4
0x00000078	USER_INT5_Handler	14	16+14: USER_INT5
0x0000007C	Spare15_Handler	15	16+ 15: Not Used
0x00000080	PORT0_0_Handler	16	16+16: GPIO Port 0 pin 0 Handler
0x00000084	PORT0_1_Handler	17	16+17: GPIO Port 0 pin 1 Handler
0x00000088	PORT0_2_Handler	18	16+18: GPIO Port 0 pin 2 Handler
0x0000008C	PORT0_3_Handler	19	16+19: GPIO Port 0 pin 3 Handler
0x00000090	PORT0_4_Handler	20	16+20: GPIO Port 0 pin 4 Handler
0x00000094	PORT0_5_Handler	21	16+21: GPIO Port 0 pin 5 Handler

Address	Interrupt	Number	Description
0x00000098	PORT0_6_Handler	22	16+22: GPIO Port 0 pin 6 Handler
0x0000009C	PORT0_7_Handler	23	16+23: GPIO Port 0 pin 7 Handler
0x000000A0	PORT0_8_Handler	24	16+24: GPIO Port 0 pin 8 Handler
0x000000A4	PORT0_9_Handler	25	16+25: GPIO Port 0 pin 9 Handler
0x000000A8	PORT0_10_Handler	26	16+26: GPIO Port 0 pin 10 Handler
0x000000AC	PORT0_11_Handler	27	16+27: GPIO Port 0 pin 11 Handler
0x000000B0	PORT0_12_Handler	28	16+28: GPIO Port 0 pin 12 Handler
0x000000B4	PORT0_13_Handler	29	16+29: GPIO Port 0 pin 13 Handler
0x000000B8	PORT0_14_Handler	30	16+30: GPIO Port 0 pin 14 Handler
0x000000BC	PORT0_15_Handler	31	16+31: GPIO Port 0 pin 15 Handler

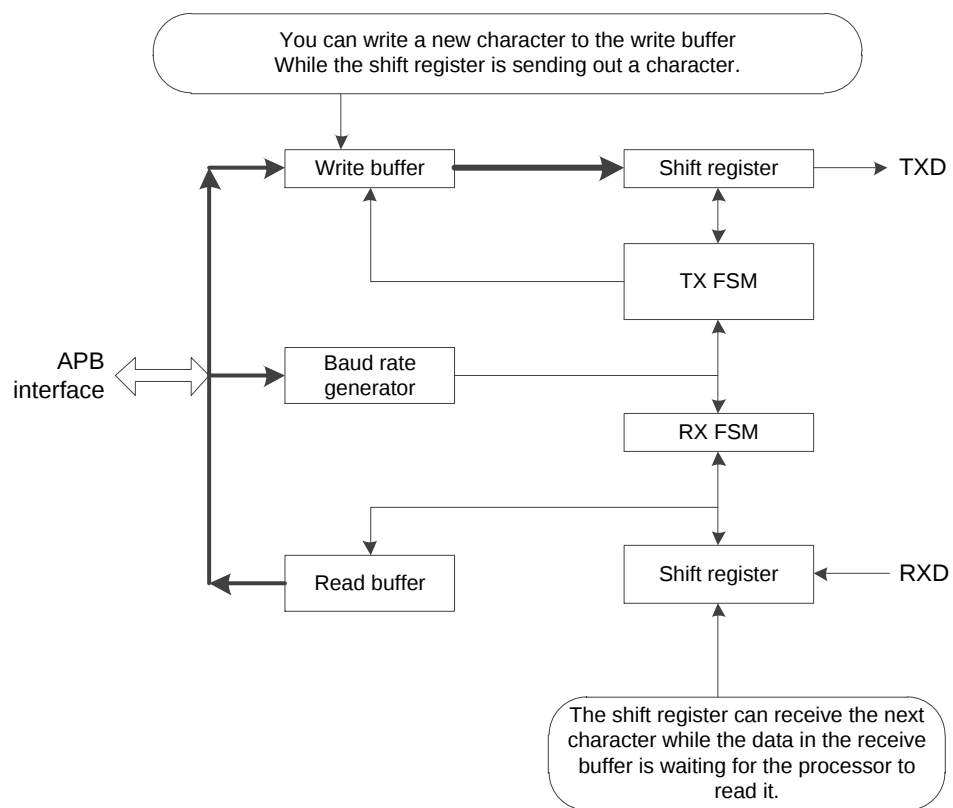
4 Universal Asynchronous Receiver/Transmitter

4.1 Features

Gowin_EMPU(GW1NS-4C) includes two UART peripherals accessed by the APB bus.

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

UART block diagram is as shown in Figure 4-1.

Figure 4-1 UART Block Diagram

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 Register Definition

The UART register definitions are as shown in Table 4-1. UART register definitions are located in library\libraries\cmsis\cm3\device_support\gw1ns4c.h.

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
INTSTATU/INT CLEAR	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 Definition

The UART initialization definitions are shown in Table 4-2. UART initialization definitions are located in library\libraries\drivers\inc\gw1ns4c_uart.h.

Table 4-2 Definition of UART Initialization

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 high speed test mode

4.4 Usage of Drivers

The usage of UART drivers is shown in Table 4-3. UART driver definitions are located in library\libraries\drivers\inc\gw1ns4c_uart.c.

Table 4-3 Usage of UART Drivers

Name	Description
UART_Init	Initializes UARTx
UART_GetRxBufferFull	Returns UARTx RX buffer full status
UART_GetTxBufferFull	Returns UARTx TX buffer full status
UART_GetRxBufferOverrunStatus	Returns UARTx RX buffer overrun status
UART_GetTxBufferOverrunStatus	Returns UARTx TX buffer overrun status
UART_ClearRxBufferOverrunStatus	Clears Rx buffer overrun status
UART_ClearTxBufferOverrunStatus	Clears Tx buffer overrun status
UART_SendChar	Sends a character to UARTx TX buffer
UART_SendString	Sends a string to UARTx TX buffer
UART_ReceiveChar	Receives a character from UARTx RX buffer
UART_GetBaudDivider	Returns UARTx baud rate divider value
UART_GetTxIRQStatus	Returns UARTx TX interrupt status
UART_GetRxIRQStatus	Returns UARTx RX interrupt status
UART_ClearTxIRQ	Clears UARTx TX interrupt status

Name	Description
UART_ClearRxIRQ	Clears UARTx RX interrupt status
UART_GetTxOverrunIRQStatus	Returns UARTx TX overrun interrupt status
UART_GetRxOverrunIRQStatus	Returns UARTx RX overrun interrupt status
UART_ClearTxOverrunIRQ	Clears UARTx TX overrun interrupt request
UART_ClearRxOverrunIRQ	Clears UARTx RX overrun interrupt request
UART_SetHSTM	Sets UARTx TX high speed test mode
UART_ClrHSTM	Clears UARTx TX high speed test mode

5 Timer

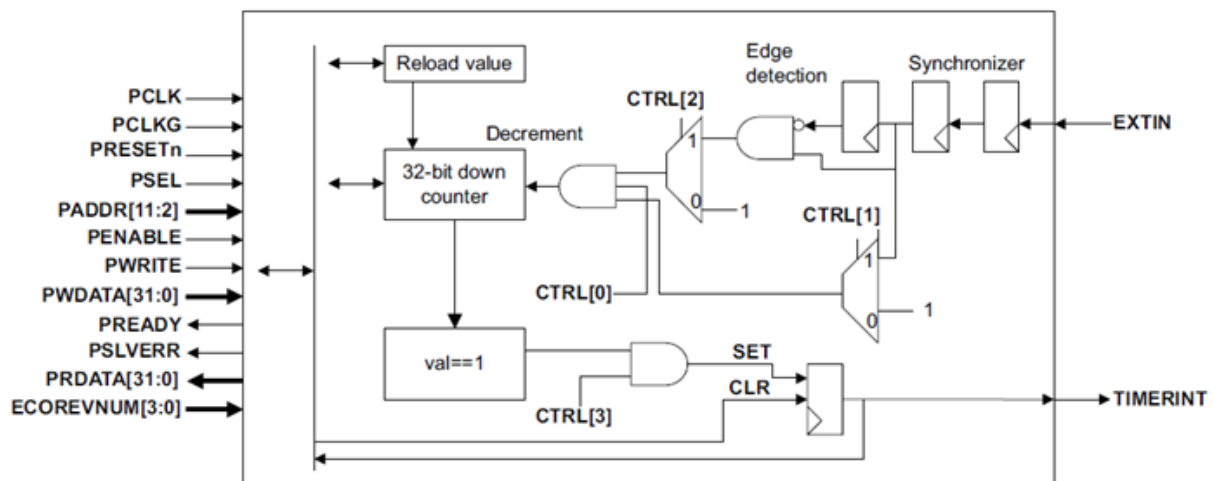
5.1 Features

Gowin_EMPU (GW1NS-4C) includes two synchronized standard timer peripherals accessed by the APB bus.

- 32-bit counter
- Supports the interrupt request signal
- Supports the external input signal EXTIN to enable clock
- Timer0: EXTIN connects to GPIO [1]
- Timer1: EXTIN connects to GPIO [6]

The TIMER block diagram is shown in Figure 5-1.

Figure 5-1 TIMER Block Diagram



5.2 Register Definition

The Timer registers definitions are as shown in Table 5-1. Timer register definitions are located in `library\libraries\cmsis\cm3\device_support\gw1ns4c.h`.

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 this register sets the current value.
INTSTATUS/INTCLEAR	0x00C	RW	1	0x0	[0] Timer interrupt, write 1 to clear

5.3 Initialization Definition

The Timer initialization definitions are as shown in Table 5-2. Timer initialization definitions are located in `library\libraries\drivers\inc\gw1ns4c_timer.h`.

Table 5-2 Definition of TIMER Initialization

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 Usage of Drivers

The usage of TIMER drivers is shown in Table 5-3. Timer driver definitions are located in library\libraries\drivers\inc\gw1ns4c_timer.c.

Table 5-3 Usage of TIMER Drivers

Name	Description
TIMER_Init	Initializes TIMERx
TIMER_StartTimer	Starts TIMERx
TIMER_StopTimer	Stops TIMERx
TIMER_GetIRQStatus	Returns TIMERx interrupt status
TIMER_ClearIRQ	Clears TIMERx interrupt status
TIMER_GetReload	Returns TIMERx reload value
TIMER_SetReload	Sets TIMERx reload value
TIMER_GetValue	Returns TIMERx current value
TIMER_SetValue	Sets TIMERx current value
TIMER_EnableIRQ	Enable TIMERx interrupt request
TIMER_DisableIRQ	Disable TIMERx interrupt request

6 Watchdog

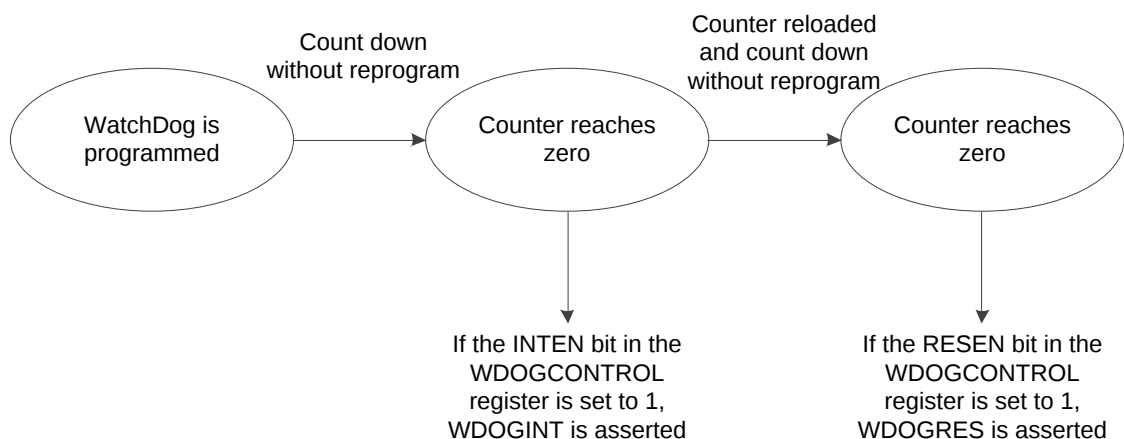
6.1 Features

Gowin_EMPU(GW1NS-4C) includes one WatchDog peripheral accessed by the APB bus.

- A 32-bit down-counter initialized by the LOAD register
- Supports interrupt request
- When the clock is enabled, the counter is decremented by the posedge of the WDOGCLK signal;
- A reset request is generated and the counter is stopped when the counter is decremented to 0 to monitor interrupts;
- Provide reset caused by software crash and method.

WatchDog operating flow is shown in Figure 6-1.

Figure 6-1 WatchDog Operating Flow



6.2 Register Definition

The WatchDog registers definitions are as shown in Table 6-1.
 WatchDog register definitions are located in
 library\libraries\cmsis\cm3\device_support\gw1ns4c.h.

Table 6-1 Definition of Watchdog Registers

Register Name	Address Offset	Type	Width	Initial Value	Description
LOAD	0x00	RW	32	0xFFFFFFFF F	The value from which the counter is to decrement
VALUE	0x04	RO	32	0xFFFFFFFF F	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 WDOGIN value

6.3 Initialization Definition

The WatchDog initialization definitions are as shown in Table 6-2. WatchDog initialization definitions are located in `library\libraries\drivers\inc\gw1ns4c_wdog.h`.

Table 6-2 Definition of WatchDog Initialization

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 Usage of Drivers

The usage of WatchDog drivers is shown in Table 6-3. WatchDog driver definitions are located in `library\libraries\drivers\inc\gw1ns4c_wdog.c`.

Table 6-3 Usage of WatchDog 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

Name	Description
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

7 GPIO

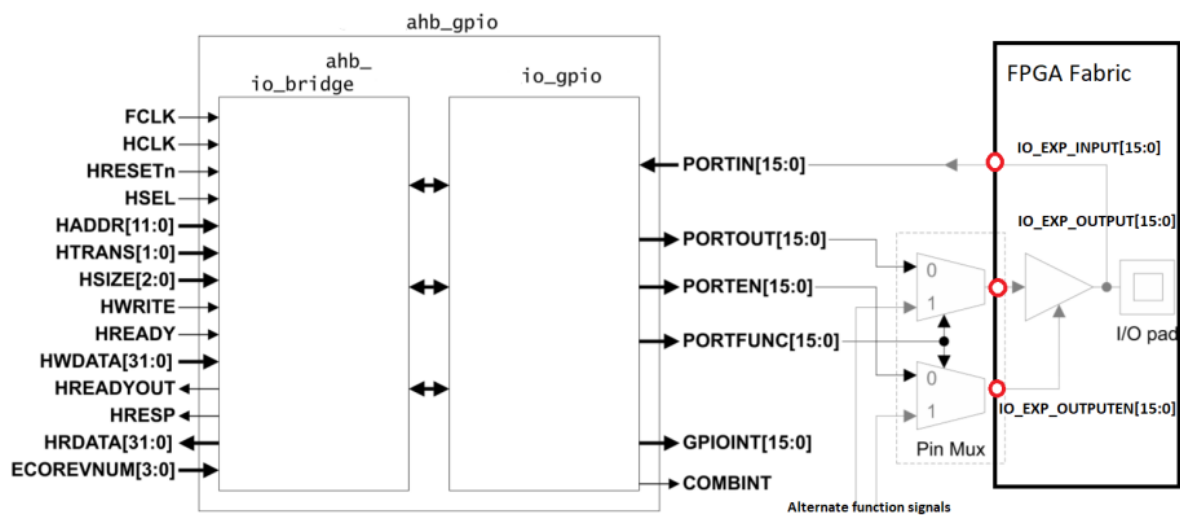
7.1 Features

Gowin_EMPU(GW1NS-4C) includes one GPIO peripheral with a 16-bit input and output interface accessed by AHB bus.

- Connects with FPGA core system
- Each IO pin can generate an interrupt;
- Supports the bit mask
- Supports pin multiplexing.

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

Figure 7-1 GPIO Block Diagram



7.2 Register Definition

The GPIO registers definitions are as shown in Table 7-1. GPIO register definitions are located in library\libraries\cmsis\cm3\device_support\gw1ns4c.h.

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 synchronization 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 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

Register Name	Address Offset	Type	Width	Initial Value	Description
					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 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 configuration 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 configuration Write 1 to clear the interrupt polarity bit

Register Name	Address Offset	Type	Width	Initial Value	Description
					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 effect Read back IRQ status register
MASKLOWBY TE	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
MASKHIGHBY TE	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 Definition

The GPIO initialization definitions are shown in Table 7-2. GPIO initialization definitions are located in `library\libraries\drivers\inc\gw1ns4c_gpio.h`.

Table 7-2 Definition of GPIO Initialization

Name	Type	Value	Description
GPIO_Pin	uint32_t	GPIO_Pin_0 GPIO_Pin_1 GPIO_Pin_2 GPIO_Pin_3	16 bits GPIO Pins

Name	Type	Value	Description
		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	
GPIO_Mode	GPIO_Mode_TypeDef	GPIO_Mode_IN GPIO_Mode_OUT GPIO_Mode_AF	16 bits GPIO Pins mode
GPIO_Int	GPIO_Int_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 Usage of Drivers

The usage of GPIO drivers is shown in Table 7-3. GPIO driver definitions are located in library\libraries\drivers\inc\gw1ns4c_gpio.c.

Table 7-3 Usage of GPIO Drivers

Name	Description
GPIO_Init	Initializes GPIOx
GPIO_SetOutEnable	Sets GPIOx output enable
GPIO_ClrOutEnable	Clears GPIOx output enable
GPIO_GetOutEnable	Returns GPIOx output enable
GPIO_SetBit	GPIO output one
GPIO_ResetBit	GPIO output zero
GPIO_WriteBits	GPIO output
GPIO_ReadBits	GPIO input

Name	Description
GPIO_SetAltFunc	Sets GPIOx alternate function enable
GPIO_ClrAltFunc	Clears GPIOx alternate function enable
GPIO_GetAltFunc	Returns GPIOx alternate function enable
GPIO_IntClear	Clears GPIOx interrupt request
GPIO_GetIntStatus	Returns GPIOx interrupt status
GPIO_SetIntEnable	Sets GPIOx interrupt enable Returns GPIOx interrupt status
GPIO_ClrIntEnable	Clears GPIOx interrupt enable Returns GPIOx interrupt enable
GPIO_SetIntHighLevel	Setups GPIOx interrupt as high level
GPIO_SetIntRisingEdge	Setups GPIOx interrupt as rising edge
GPIO_SetIntLowLevel	Setups GPIOx interrupt as low level
GPIO_SetIntFallingEdge	Setups GPIOx interrupt as falling edge
GPIO_MaskedWrite	Setups GPIOx output value using masked access

8 Real-time Clock

8.1 Features

Gowin_EMPU(GW1NS-4C) includes one 32-bit real-time clock (RTC) peripheral accessed by APB bus.

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

MCU reads and writes data from/to RTC, control RTC, and read RTC status via APB bus interface. The 32-bit counter increases on the rising edge of the continuous input clock CLK1HZ (rtc_src_clk receives 3.072MHz clock input, and RTC internal division is 1Hz).

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 go back to 0 and keep increasing.

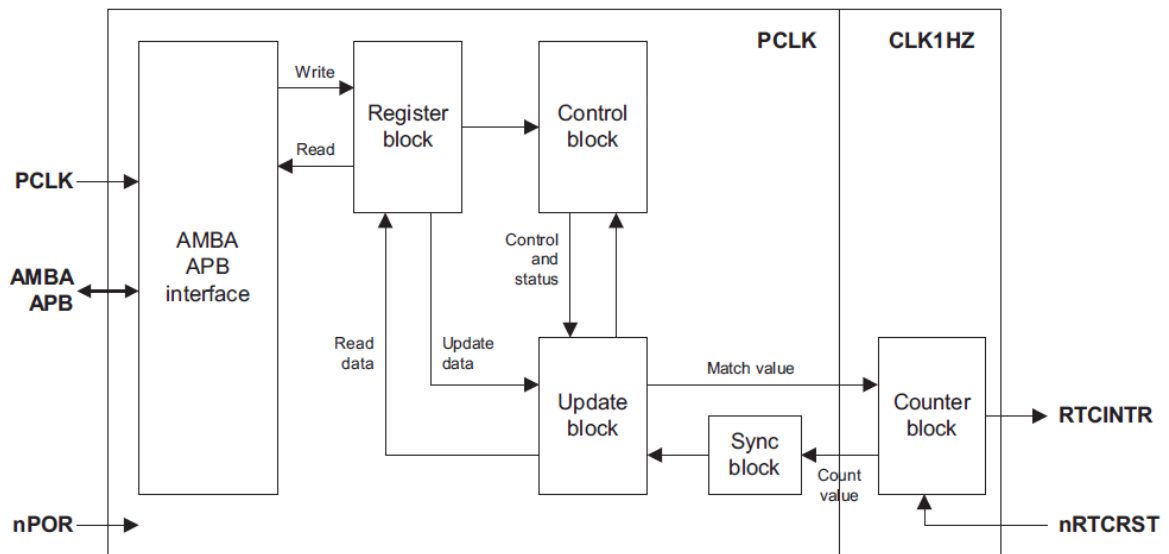
Realizes RTC load or update via the write load register
RTC_LOAD_VALUE

Obtains RTC current clock via the read data register
RTC_CURRENT_DATA

Programs Match register via the register of write
RTC_MATCH_VALUE

The RTC block diagram is shown in Figure 8-1.

Figure 8-1 RTC Block Diagram



8.2 Register Definition

The RTC registers definitions are shown in Table 8-1. RTC register definitions are located in
library\libraries\cmsis\cm3\device_support\gw1ns4c.h.

Table 8-1 Definition of RTC 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_COUNTER_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

Register Name	Address Offset	Type	Width	Initial Value	Description
					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 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

8.3 Usage of Drivers

The usage of RTC drivers is shown in Table 8-2. RTC driver definitions are located in library\libraries\drivers\inc\gw1ns4c_rtc.c.

Table 8-2 Usage of RTC Drivers

Name	Description
RTC_init	Initializes RTC
Get_Current_Value	Gets RTC current value of data register
Set_Match_Value	Sets RTC match value of match register
Get_Match_Value	Gets RTC match value of match register
Set_Load_Value	Sets RTC load value of load register
Get_Load_Value	Gets RTC load value of load register
Start_RTC	Starts RTC counter
Close_RTC	Closes RTC counter
RTC_Inter_Mask_Set	Sets RTC interrupt mask

Name	Description
Get_RTC_Control_value	Gets value of control register
RTC_Inter_Mask_Clr	Clears RTC interrupt mask
Get_RTC_Inter_Mask_value	Gets RTC interrupt mask
Clear_RTC_interrupt	Clears RTC interrupt

9 SPI

9.1 Features

Gowin_EMPU(GW1NS-4C) includes one SPI Master peripheral accessed by APB bus.

- APB 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 bit width for data RX register and data TX register

9.2 Register Definition

The definitions of SPI Master registers are as shown in Table 9-1. SPI Master register definitions are located in library\libraries\cmsis\cm3\device_support\gw1ns4c.h.

Table 9-1 Definition of SPI Master Registers

Register Name	Address Offset	Type	Width	Initial Value	Description
RDATA	0x00	RO	8	0x00	Reads data register
WDATA	0x04	WO	8	0x00	Writes data register
STATUS	0x08	RW	8	0x00	[7] Error status [6] Receives ready status [5] Transmits ready status [4] Be transmitting [3] Transmits overrun error status [2] Receives overrun error status

Register Name	Address Offset	Type	Width	Initial Value	Description
					[1:0] Reserved
SSMASK	0x0C	RW	8	0x00	Unused selected slave address
CTRL	0x10	RW	5	0x00	[4:3] Clock selected [2] Polarity [1] Phase [0] Direction

9.3 Initialization Definition

The SPI Master initialization definitions are shown in Table 9-2. SPI Master initialization definitions are located in `library\libraries\drivers\inc\gw1ns4c_spi.h`.

Table 9-2 Definition of SPI Master Initialization

Name	Type	Value	Description
DIRECTION	uint8_t	1/0	MSB/LSB first transmission 0: MSB first; 1: LSB first.
PHASE	uint8_t	1/0	Posedge/Negedge transmit data 0: Sample at posedge edge; 1: Sample at negedge edge.
POLARITY	uint8_t	1/0	Initialize polarity to one/zero 0: Idle sclk low; 1: Idle sclk high.
CLKSEL	uint8_t	CLKSEL_CLK_DIV_2 CLKSEL_CLK_DIV_4 CLKSEL_CLK_DIV_6 CLKSEL_CLK_DIV_8	Select clock divided 2/4/6/8 $CLKDIV = (CLKSEL + 1) * 2$

9.4 Usage of Drivers

The usage of SPI Master drivers is shown in Table 9-3. SPI Master driver definitions are located in `library\libraries\drivers\inc\gw1ns4c_spi.c`.

Table 9-3 Usage of SPI Master Drivers

Name	Description
SPI_Init	Initializes SPI Master
SPI_SetDirection	Sets direction
SPI_ClrDirection	Clears direction

Name	Description
SPI_GetDirection	Returns direction
SPI_SetPhase	Sets phase
SPI_ClrPhase	Clears phase
SPI_GetPhase	Returns phase
SPI_SetPolarity	Sets polarity
SPI_ClrPolarity	Clears polarity
SPI_GetPolarity	Returns polarity
SPI_SetClkSel	Sets clock selection
SPI_GetClkSel	Returns clock selection
SPI_GetToeStatus	Reads transmit overrun error status
SPI_GetRoeStatus	Reads receive overrun error status
SPI_GetTmtStatus	Reads transmitting status
SPI_GetTrdyStatu	Reads transmit ready status
SPI_GetRrdyStatus	Reads receive ready error status
SPI_GetErrStatus	Reads error status
SPI_ClrToeStatus	Clears transmit overrun error status
SPI_ClrRoeStatus	Clears eceive overrun error status
SPI_ClrErrStatus	Clears error status
SPI_ReadWriteByte	Full duplex read and write a byte
SPI_WriteData	Writes data
SPI_ReadData	Reads data
SPI_Select_Slave	Selects slave

10 System Controller

10.1 Register Definition

The SYSCON register definitions are shown in Table 10-1. SYSCON register definitions are located in library\libraries\cmsis\cm3\device_support\gw1ns4c.h.

Table 10-1 Definition of SYSCON Register

Register Name	Address Offset	Type	Width	Initial Value	Description
REMAP	0x000	RW	1	0x0	Remap control register
PMUCTRL	0x004	RW	1	0x0	PMU control register
RESETOP	0x008	RW	1	0x0	reset option register
RESERVED0	0x00C	—	—	—	Reserved
RSTINFO	0x010	RW	3	0x0	[2] Lockup reset [1] Watchdog reset request [0] System reset request

10.2 Usage of Drivers

The usage of SYSCON drivers is shown in Table 10-2. SYSCON driver definitions are located in library\libraries\drivers\inc\gw1ns4c_syscon.c.

Table 10-2 Usage of SYSCON Drivers

Name	Description
SYSCON_Init	Initializes SYSCON
SYSCON_GetRemap	Returns REMAP
SYSCON_GetPmuctrlEnable	Returns PMUCTRL Enable
SYSCON_GetResetopLockuprst	Returns RESETOP LOCKUPRST
SYSCON_GetRstinfoSysresetreq	Returns RSTINFO SYSRESETREQ

Name	Description
SYSCON_GetRstinfoWdogresetreq	Returns RSTINFO SYSRESETREQ
SYSCON_GetRstinfoLockreset	Returns RSTINFO SYSRESETREQ

11 I²C Master

11.1 Features

Gowin_EMPU(GW1NS-4C) includes one I²C Master peripheral accessed by the APB bus.

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

11.2 Register Definition

The definitions of I²C Master registers are as shown in Table 11-1. I²C Master register definitions are located in library\libraries\cm3\device_support\gw1ns4c.h.

Table 11-1 Definition of I²C Master Registers

Register Name	Address Offset	Type	Width	Initial Value	Description
PRER	0x00	RW	16	0xFFFF	Clock prescale register [15:0] Prescale value = $\text{sys_clk}/(5 \times \text{SCL}) - 1$
CTR	0x04	RW	8	0x00	[7] Enable I ² C function [6] Enable I ² C interrupt [5:0] Reserved

Register Name	Address Offset	Type	Width	Initial Value	Description
TXR	0x08	WO	8	0x00	[7:1] Next transmission data [0] Data direction
RXR	0x0C	RO	8	0x00	[7:0] Last received data
CR	0x010	WO	8	0x00	[7] Start transmission status [6] Over transmission status [5] Read enable, read data from slave [4] Write enable, write data to slave [3] Acknowledge [2:1] Reserved [0] Interrupt acknowledge
SR	0x14	RO	8	0x00	[7] Receive acknowledge signal from slave [6] I ² C busy status [5] Arbitration loss [4:2] Reserved [1] Data transmission status flag [0] Interrupt flag

11.3 Usage of Drivers

The usage of I²C Master drivers is shown in Table 11-2. I²C Master driver definitions are located in library\libraries\drivers\inc\gw1ns4c_i2c.c.

Table 11-2 Usage of I²C Master Drivers

Name	Description
I2C_Init	I ² C Initialization
I2C_SendByte	Sends a byte to I ² C bus
I2C_SendBytes	Sends multiple bytes to I ² C bus
I2C_SendData	Sends multiple bytes to I ² C bus once time
I2C_ReceiveByte	Reads a byte from I ² C bus
I2C_ReadBytes	Reads multiple bytes from I ² C bus
I2C_ReceiveData	Reads multiple bytes from I ² C bus once time
I2C_Rate_Set	Sets I ² C traffic rate
I2C_Enable	Enable I ² C bus

Name	Description
I2C_UnEnable	Disable I ² C bus
I2C_InterruptOpen	Opens I ² C interrupt
I2C_InterruptClose	Closes I ² C interrupt

12 SysTick

12.1 Features

Gowin_EMPU(GW1NS-4C) includes one 24-bit system tick timer SysTick with the function of automatic overload and overrun interrupt to get certain time interval through this timer.

12.2 Register Definition

The SysTick registers definitions are shown in Table 12-1. SysTick register definitions are located in library\libraries\cmsis\cm3\core_support\mdk(gmd)\core_cm3.h.

Table 12-1 Definition of SysTick Registers

Register Name	Address Offset	Type	Width	Initial Value	Description
CTRL	0x00	RW	17	0x00000	[16] Counts down to zero flag [2] External clock source [1] Enable interrupt request [0] Enable SysTick
LOAD	0x04	RW	24	0x000000	[23:0] Reloads value
VAL	0x08	RW	24	0x000000	[23:0] Returns current count down value
CALIB	0x0C	RW	32	0x00000000	[31] Whether a separate reference clock is provided [30] Whether the TENMS value is exact [23:0] 10ms calibration value

12.3 Usage of Drivers

The usage of SysTick drivers is shown in Table 12-2. SysTick driver definitions are located in library\libraries\drivers\inc\gw1ns4c_misc.c.

Table 12-2 Usage of SysTick Drivers

Name	Description
SysTick_Config	Initializes and starts the SysTick counter and interrupt
SysTick_CLKSourceConfig	Configure the SysTick clock source

13 Memory Management

13.1 Features

Gowin_EMPU (GW1NS-4C) supports dynamic memory management and re-defines malloc and free functions to realize dynamic memory application and release.

13.2 Usage of Driver

The usage of memory management drivers is shown in Table 13-1. Memory management driver definitions are located in library\middlewares\dmm\malloc.c.

Table 13-1 Usage of Memory Management Drivers

Name	Description
mem_init	Initializes memory management
mymemset	Sets the values of memory space
mymemcpy	Copies from source to destination
mymemcmp	Compares source with destination
mymalloc	Allocates a memory space dynamically
myfree	Frees a memory space

14 SPI-Flash Memory

14.1 Features

Gowin_EMPU(GW1NS-4C) includes one SPI-Flash Memory peripheral accessed by the APB bus.

- Serial non-volatile flash
- Supports APB interface
- Supports standard SPI interface
- Supports read, write and erase functions
- Supports FPGA products GW1NSR-4C/GW1NSER-4C QN48G

14.2 Register Definition

The SPI-Flash Memory register definitions are shown in Table 14-1. SPI-Flash Memory register definitions are located in library\middlewares\spi_flash\spi_flash.h.

Table 14-1 Definition of SPI-Flash Memory Registers

Register 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

Register Name	Address Offset	Type	Width	Initial Value	Description
					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 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
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

Register Name	Address Offset	Type	Width	Initial Value	Description
					[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 [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)

Register Name	Address Offset	Type	Width	Initial Value	Description
					[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 address phase for SPI read transfers

Register Name	Address Offset	Type	Width	Initial Value	Description
					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 in 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

Register Name	Address Offset	Type	Width	Initial Value	Description
					[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 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

Register Name	Address Offset	Type	Width	Initial Value	Description
					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 [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

Register Name	Address Offset	Type	Width	Initial Value	Description
					[8] 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

14.3 Usage of Drivers

The usage of SPI-Flash Memory drivers is shown in Table 14-2. SPI-Flash Memory driver definitions are located in library\middlewares\spi_flash\spi_flash.c.

Table 14-2 Usage of SPI-Flash Memory Drivers

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

15 PSRAM Memory Interface

15.1 Features

Gowin_EMPU(GW1NS-4C) includes one PSRAM Memory Interface peripheral accessed by the AHB bus.

- Supports AHB interface
- Compatible with standard PSRAM device interface
- Supports GW1NSR-4C MG64P device
- Supports 8-bit PSRAM width
- Supports 16-bit DQ bus width
- Supports programmable burst length 16, 32, 64, 128
- Clock ratio of 1:2
- Supports initial delay of 6
- Supports fixed delay mode
- Supports power off option
- Configurable drive strength
- Configurable self-refreshing area
- Configurable refresh rate

15.2 Register Definition

The PSRAM Memory Interface registers definitions are shown in Table 15-1. PSRAM Memory Interface register definitions are located in library\middlewarees\psram\psram.h.

Table 15-1 Definition of PSRAM Memory Interface Registers

Register 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 32-bit data
RD_DATA1	0x24	RO	32	-	Read data register 1 [31:0] Read second 32-bit data
RD_DATA2	0x28	RO	32	-	Read data register 2 [31:0] Read third 32-bit data
RD_DATA3	0x2C	RO	32	-	Read data register 3 [31:0] Read fourth 32bit data

Register Name	Address Offset	Type	Width	Initial Value	Description
INTI_DONE	0x30	RO	1	-	Initialization done register [0] PSRAM hardware initialization done flag 0 = Initialization failed 1 = Initialization done

15.3 Usage of Drivers

The usage of PSRAM Memory Interface drivers is shown in Table 15-2. PSRAM Memory Interface driver definitions are located in library\middlewarees\psram\psram.c.

Table 15-2 Usage of PSRAM Memory Interface Drivers

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

16 HyperRAM Memory Interface

16.1 Features

Gowin_EMPU(GW1NS-4C) includes one HyperRAM Memory Interface peripheral accessed by the AHB bus.

- Supports AHB interface
- Compatible with standard HyperRAM device interface
- Supports GW1NSR-4C/GW1NSER-4C QN48P device
- Supports 8-bit PSRAM width
- Supports 8-bit DQ bus width
- Supports programmable burst length 16, 32, 64, 128
- Clock ratio of 1:2
- Supports initial delay of 6
- Supports fixed delay mode
- Supports power off option
- Configurable drive strength
- Configurable self-refreshing area
- Configurable refresh rate

16.2 Register Definition

The HyperRAM Memory Interface registers definitions are shown in Table 16-1. HyperRAM Memory Interface register definitions are located in library\middlewarees\hyper_ram\hyper_ram.h.

Table 16-1 Definition of HyperRAM Memory Interface Registers

Register 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 HyperRAM Memory
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

Register Name	Address Offset	Type	Width	Initial Value	Description
					[0] HyperRAM Memory hardware initialization done flag 0 = Initialization failed 1 = Initialization done

16.3 Usage of Drivers

The usage of HyperRAM Memory Interface drivers is shown in Table 16-2. HyperRAM Memory Interface driver definitions are located in `library\middlewares\hyper_ram\hyper_ram.c`.

Table 16-2 Usage of HyperRAM Memory Interface Drivers

Name	Description
HPRAM_Check_Init_Status	Check the status of HyperRAM Memory initialization
HPRAM_Mode_Set	Set the mode for HyperRAM Memory write and read
HPRAM_Address_Set	Set the address of HyperRAM Memory and save data into this address
HPRAM_Read_Data_Buff	Read data from the buffer of HyperRAM Memory
HPRAM_Cmd_Enable	Enable the command of HyperRAM Memory
HPRAM_Read_Done_Flag	Get the flag of read HyperRAM Memory done
HPRAM_Clear_Read_Done_Flag	Clear the flag of read HyperRAM Memory done
HPRAM_Write_Data_Buff	Write data into the buffer of HyperRAM Memory
HPRAM_Cmd_Unable	Disable the command of HyperRAM Memory
HPRAM_Write_Data_Package	Write a package data into HyperRAM Memory
HPRAM_Read_Data_Package	Read a package data from HyperRAM Memory

17 Real-time Operating System

Gowin_EMPU(GW1NS-4C) supports real-time operating system, such as uC/OS-III, FreeRTOS, and RT-Thread Nano version.

17.1 uC/OS-III

17.1.1 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(GW1NS-4C) supports uC/OS-III.
- uC/OS-III source code is available at Micrium website: <http://www.micrium.com>

17.1.2 Version

Gowin_EMPU(GW1NS-4C) supports uC/OS-III V3.03.00.

17.1.3 Configuration

- You can modify UCOSIII_CONFIG\os_cfg.h and os_cfg_app.h to configure uC/OS-III.
- You can modify UCOS_BSP\bsp.c and bsp.h to support the development board used.

17.2 FreeRTOS

17.2.1 Features

- FreeRTOS is lightweight real-time operating system.
- As a lightweight operating system, FreeRTOS 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(GW1NS-4C) supports FreeRTOS.
- FreeRTOS source code is available at FreeRTOS website: <http://www.FreeRTOS.org>

17.2.2 Version

Gowin_EMPU(GW1NS-4C) supports FreeRTOS V10.2.1.

17.2.3 Configuration

You can modify include\FreeRTOSConfig.h to configure FreeRTOS.

17.3 RT-Thread Nano Version

17.3.1 Features

- RT-Thread Nano is a lite hard real-time core, developed in C language with object-oriented programming ideas, and it is a trimmable, preemptive real-time multitasking RTOS with a good code style.
- Its memory resource utilization is extremely small and its functions include task processing, software timers, semaphores, mailboxes and real-time scheduling, etc.
- It is free and open source and follows the Apache license 2.0. RTOS core and all open source components are free to use in commercial products without the need to publish application source code and without potential commercial risk.
- Gowin_EMPU(GW1NS-4C) supports RT-Thread Nano version.
- RT-Thread Nano source code is available at the RT-Thread website <https://www.rt-thread.org>.

17.3.2 Version

Gowin_EMPU(GW1NS-4C) supports RT-Thread Nano V3.1.5.

17.3.3 Configuration

You can modify `bsp\cm3\rtconfig.h` to configure RT-Thread Nano.

You can modify `bsp\cm3\drivers\board.c` to support the development board used.

18 Application Programs

Gowin_EMPU (GW1NS-4C) provides application programs for ARM Keil MDK IDE (tested software version: V5.26) and GMD IDE (tested software version: V1.2) software environment. Click the following link to get the application program:

[cdn.gowinsemi.com.cn/Gowin_EMPU\(GW1NS-4C\)_V2.0.zip](http://cdn.gowinsemi.com.cn/Gowin_EMPU(GW1NS-4C)_V2.0.zip)

18.1 UART

Gowin_EMPU(GW1NS-4C) provides UART application program:

...\ref_design\MCU_RefDesign\MDK_RefDesign\cm3_demo\project\p
rintf

...\ref_design\MCU_RefDesign\GMD_RefDesign\cm3_demo\src\proje
ct\printf

...\ref_design\MCU_RefDesign\MDK_RefDesign\cm3_demo\project\u
art_rx

...\ref_design\MCU_RefDesign\GMD_RefDesign\cm3_demo\src\proje
ct\uart_rx

...\ref_design\MCU_RefDesign\MDK_RefDesign\cm3_demo\project\u
art_rx_intr

...\ref_design\MCU_RefDesign\GMD_RefDesign\cm3_demo\src\proje
ct\uart_rx_intr

18.2 Timer

Gowin_EMPU(GW1NS-4C) provides Timer application program:

...\ref_design\MCU_RefDesign\MDK_RefDesign\cm3_demo\project\ti
mer

```
...\\ref_design\\MCU_RefDesign\\GMD_RefDesign\\cm3_demo\\src\\project\\timer
```

18.3 Watch Dog

Gowin_EMPU(GW1NS-4C) provides Watch Dog application program:

```
...\\ref_design\\MCU_RefDesign\\MDK_RefDesign\\cm3_demo\\project\\wdog
```

```
...\\ref_design\\MCU_RefDesign\\GMD_RefDesign\\cm3_demo\\src\\project\\wdog
```

18.4 GPIO

Gowin_EMPU(GW1NS-4C) provides GPIO application program:

```
...\\ref_design\\MCU_RefDesign\\MDK_RefDesign\\cm3_demo\\project\\led
```

```
...\\ref_design\\MCU_RefDesign\\GMD_RefDesign\\cm3_demo\\src\\project\\led
```

```
...\\ref_design\\MCU_RefDesign\\MDK_RefDesign\\cm3_demo\\project\\gpio_in_intr
```

```
...\\ref_design\\MCU_RefDesign\\GMD_RefDesign\\cm3_demo\\src\\project\\gpio_in_intr
```

```
...\\ref_design\\MCU_RefDesign\\MDK_RefDesign\\cm3_demo\\project\\keyscan
```

```
...\\ref_design\\MCU_RefDesign\\GMD_RefDesign\\cm3_demo\\src\\project\\keyscan
```

18.5 I²C Master

Gowin_EMPU(GW1NS-4C) provides I²C Master application program:

```
...\\ref_design\\MCU_RefDesign\\MDK_RefDesign\\cm3_demo\\project\\i2c_master
```

```
...\\ref_design\\MCU_RefDesign\\GMD_RefDesign\\cm3_demo\\src\\project\\i2c_master
```

18.6 SPI Master

Gowin_EMPU(GW1NS-4C) provides SPI Master application program:

```
...\\ref_design\\MCU_RefDesign\\MDK_RefDesign\\cm3_demo\\project\\spi_master
...\\ref_design\\MCU_RefDesign\\GMD_RefDesign\\cm3_demo\\src\\project\\spi_master
```

18.7 RTC

Gowin_EMPU(GW1NS-4C) provides RTC application program:

```
...\\ref_design\\MCU_RefDesign\\MDK_RefDesign\\cm3_demo\\project\\rtc
...\\ref_design\\MCU_RefDesign\\GMD_RefDesign\\cm3_demo\\src\\project\\rtc
```

18.8 SPI-Flash Memory

Gowin_EMPU(GW1NS-4C) provides SPI-Flash Memory application program:

```
...\\ref_design\\MCU_RefDesign\\MDK_RefDesign\\cm3_demo\\project\\spi_flash
...\\ref_design\\MCU_RefDesign\\GMD_RefDesign\\cm3_demo\\src\\project\\spi_flash
```

18.9 PSRAM Memory

Gowin_EMPU(GW1NS-4C) provides PSRAM Memory application program:

```
...\\ref_design\\MCU_RefDesign\\MDK_RefDesign\\cm3_demo\\project\\psram
...\\ref_design\\MCU_RefDesign\\GMD_RefDesign\\cm3_demo\\src\\project\\psram
```

18.10 HyperRAM Memory

Gowin_EMPU(GW1NS-4C) provides HyperRAM Memory application program:

```
...\\ref_design\\MCU_RefDesign\\MDK_RefDesign\\cm3_demo\\project\\hyper_ram
...\\ref_design\\MCU_RefDesign\\GMD_RefDesign\\cm3_demo\\src\\project\\hyper_ram
```

18.11 Interrupt

Gowin_EMPU(GW1NS-4C) provides Interrupt application program:

```
...\ref_design\MCU_RefDesign\MDK_RefDesign\cm3_demo\project\interrupt
...\ref_design\MCU_RefDesign\GMD_RefDesign\cm3_demo\src\project\interrupt
```

18.12 DMM

Gowin_EMPU(GW1NS-4C) provides Dynamic Memory Management application program:

```
...\ref_design\MCU_RefDesign\MDK_RefDesign\cm3_demo\project\dmm
...\ref_design\MCU_RefDesign\GMD_RefDesign\cm3_demo\src\project\dmm
```

18.13 AHB Master

Gowin_EMPU(GW1NS-4C) provides AHB Master application program:

```
...\ref_design\MCU_RefDesign\MDK_RefDesign\cm3_demo\project\ahb2_master
...\ref_design\MCU_RefDesign\GMD_RefDesign\cm3_demo\src\project\ahb2_master
```

18.14 APB Master

Gowin_EMPU(GW1NS-4C) provides APB Master application program:

```
...\ref_design\MCU_RefDesign\MDK_RefDesign\cm3_demo\project\apb2_master
...\ref_design\MCU_RefDesign\GMD_RefDesign\cm3_demo\src\project\apb2_master
```

18.15 uC/OS-III

Gowin_EMPU(GW1NS-4C) provides RTOS uC/OS-III application program:

...\ref_design\MCU_RefDesign\MDK_RefDesign\cm3_ucos_iii

...\ref_design\MCU_RefDesign\GMD_RefDesign\cm3_ucos_iii

18.16 FreeRTOS

Gowin_EMPU(GW1NS-4C) provides RTOS FreeRTOS application program:

...\ref_design\MCU_RefDesign\MDK_RefDesign\cm3_freertos

...\ref_design\MCU_RefDesign\GMD_RefDesign\cm3_freertos

18.17 RT-Thread Nano Version

Gowin_EMPU(GW1NS-4C) provides RT-Thread Nano version application program:

...\ref_design\MCU_RefDesign\MDK_RefDesign\cm3_rtthread_nano

...\ref_design\MCU_RefDesign\GMD_RefDesign\cm3_rtthread_nano

