Gowin_EMPU_M1

Software Programming Reference Manual

IPUG533-1.0E,2/19/2019

Disclaimer

Revision History

Date	Version	Description
2/19/2019	1.0E	Initial version published.

Contents

List of Figures

List of Tables

1 Software Programming Libraries

Gowin_EMPU_M1 software programming libraries provided by Gowin are as follows:

- Gowin_EMPU_M1_Src\c_lib
- Two Gowin_EMPU_M1 software programming methods are supported:
- Cortex-M1 software programming
- Embedded Operation System software programming

1.1 Cortex-M1 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 Software Programming

File	Description
startup_GOWIN_M1.s	Cortex-M1startup program
core_cm1.h	CORE, Cortex-M1
GOWIN_M1.h	Definition of interrupt vector table, register, and address mapping
system_GOWIN_M1.c	Cortex-M1 system initialization and system clock definition
GOWIN_M1_flash.ld	Flash linker script in GNU tool chain
GOWIN_M1_gpio.c	Definition of GPIO driving function
GOWIN_M1_timer.c	Definition of Timer driving function
GOWIN_M1_wdog.c	Definition of Watch Dog driving function
GOWIN_M1_uart.c	Definition of UART driving function
GOWIN_M1_misc.c	Interrupt priority management and SysTick
GOWIN_M1_it.c	Definition of interrupt handling function

1.2 Embedded Operation System Software Programming

Two Gowin_EMPU_M1 operation system software programming methods are supported:

- uC/OS-III
- FreeRTOS

2 Memory System

2.1 Standard Peripherals Memory Mapping

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

Table 2-1 Memory Mapping of Gowin_EMPU_M1 Standard Peripherals

Standard Peripheral	Type	Address Mapping	Description
ITCM		0x00000000	1KB, 2KB, 4KB, 8KB, 16KB, 32KB
DTCM		0x20000000	1KB, 2KB, 4KB, 8KB, 16KB, 32KB
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
GPIO0	GPIO_TypeDef	0x40000000	GPIO port
APB2 Extension		0xB0000000	APB Extension interface
AHB2 Extension		0xA0000000	AHB Extension interface

2.2 Memory Mapping of Core System

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

Table 2-2 Cortex-M1 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

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

- Provides 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

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

Table 3-1 Cortex-M1 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: TIMER 0 handler
0x0000004C	TIMER1_Handler	3	16+ 3: TIMER 1 handler
0x00000050	GPIO0_Handler	4	16+ 4: GPIO Port 0 Combined Handler
0x00000054	UARTOVF_Handler	5	16+ 5: UART 0,1 Overflow Handler
0x00000058	Interrupt6_Handler	6	16+ 6: Interrupt 6 Handler
0x0000005C	Interrupt7_Handler	7	16+ 7: Interrupt 7 Handler
0x00000060	Interrupt8_Handler	8	16+ 8: Interrupt 8 Handler
0x00000064	Interrupt9_Handler	9	16+ 9: Interrupt 9 Handler
0x00000068	Interrupt10_Handler	10	16+10: Interrupt 10 Handler
0x0000006C	Interrupt11_Handler	11	16+11: Interrupt 11 Handler

Address	Interrupt	Number	Description
0x00000070	Interrupt12_Handler	12	16+12: Interrupt 12 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	Interrupt16_Handler	16	16+16: Interrupt 16 Handler
0x00000084	Interrupt17_Handler	17	16+17: Interrupt 17 Handler
0x00000088	Interrupt18_Handler	18	16+18: Interrupt 18 Handler
0x0000008C	Interrupt19_Handler	19	16+19: Interrupt 19 Handler
0x00000090	Interrupt20_Handler	20	16+20: Interrupt 20 Handler
0x00000094	Interrupt21_Handler	21	16+21: Interrupt 21 Handler
0x00000098	Interrupt22_Handler	22	16+22: Interrupt 22 Handler
0x0000009C	Interrupt23_Handler	23	16+23: Interrupt 23 Handler
0x000000A0	Interrupt24_Handler	24	16+24: Interrupt 24 Handler
0x000000A4	Interrupt25_Handler	25	16+25: Interrupt 25 Handler
0x000000A8	Interrupt26_Handler	26	16+26: Interrupt 26 Handler
0x000000AC	Interrupt27_Handler	27	16+27: Interrupt 27 Handler
0x000000B0	Interrupt28_Handler	28	16+28: Interrupt 28 Handler
0x000000B4	Interrupt29_Handler	29	16+29: Interrupt 29 Handler
0x000000B8	Interrupt30_Handler	30	16+30: Interrupt 30 Handler
0x000000BC	Interrupt31_Handler	31	16+31: Interrupt 31 Handler

4 Universal Asynchronous Receiver/Transmitter

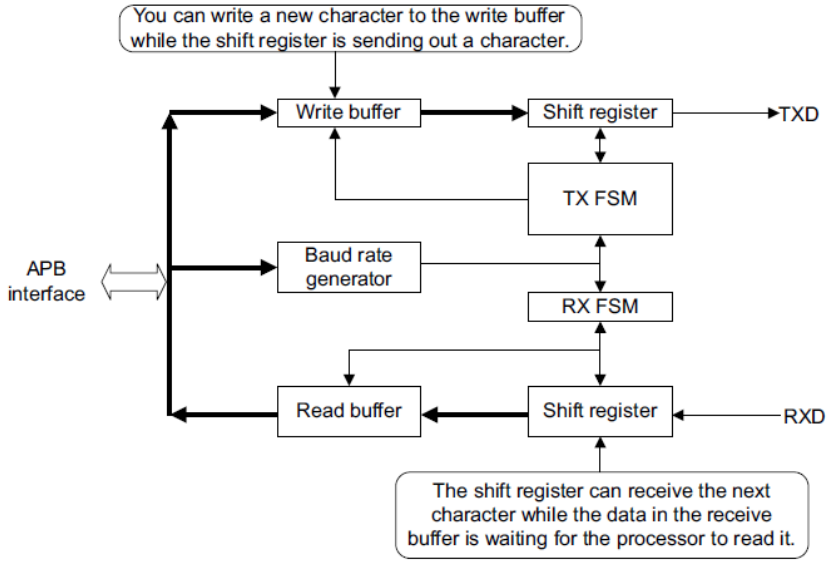
4.1 Features

Gowin_EMPU_M1 contains two UART accessed by APB1 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 a cycle, and the text information can be transmitted in a short time.

The baud rate divider register should be set in 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 definition is as shown in Table 4-1.

Table 4-1 UART Register Definition

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 Usage of Function Library

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

Table 4-3 Usage of UART Function Library

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

Name	Description
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
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

4.5 Reference Design

Gowin provides UART reference design for Keil and GNU software environment:

- MCU_RefDesign\Keil_RefDesign\uart
- MCU_RefDesign\GNU_RefDesign\m1_uart

5 Timer

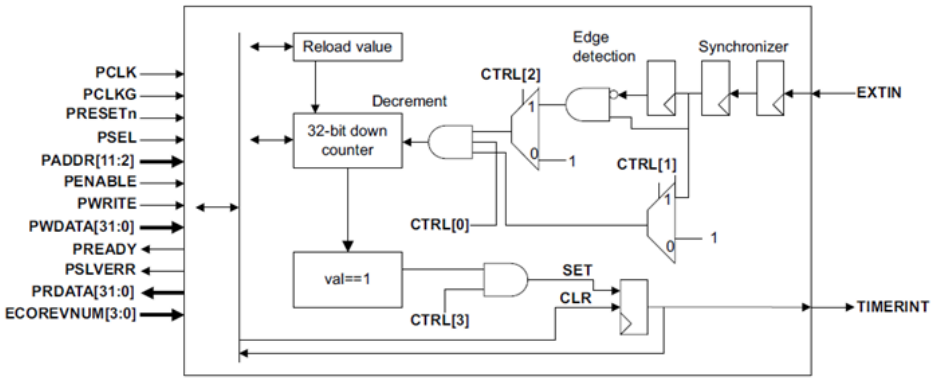
5.1 Features

Gowin_EMPU_M1 contains two synchronization standard timers accessed by the APB1 bus:

- 32-bit counter
- Supports the interrupt request signal
- Supports the external input signal EXTIN enable clock
- TIMER0: EXTIN connects GPIO [1]
- TIMER1: EXTIN connects GPIO [6]

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

Figure 5-1 Timer



5.2 Register Definition

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

Table 5-1 Definition of Timer Register

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

Register Name	Address Offset	Type	Width	Initial Value	Description
INTSTATUS/INT CLEAR	0x00C	RW	1	0x0	[0] Timer interrupt,write 1 to clear

5.3 Initialization Definition

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

Table 5-2 Timer Initialization Structure

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 Function Library

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

Table 5-3 Usage of Timer Function Library

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

5.5 Reference Design

Gowin provides Timer reference design for Keil and GNU software environment:

- MCU_RefDesign\Keil_RefDesign\timer
- MCU_RefDesign\GNU_RefDesign\m1_timer

6 Watchdog

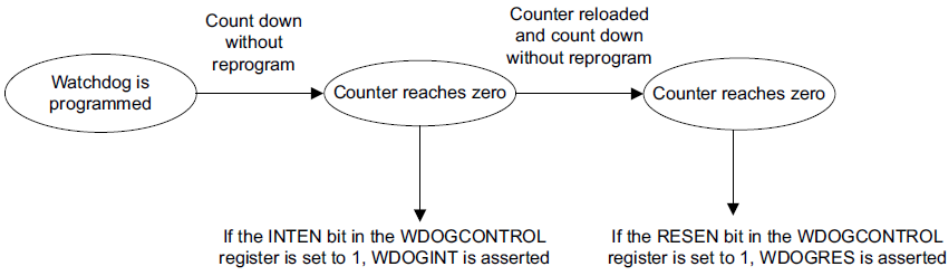
6.1 Features

Gowin_EMPU_M1 contains one Watch Dog accessed by APB1bus:

- A 32-bit down-counter initialized by the LOAD register
- Supports interrupt request
- When the clock is enabled, the counter is decremented by the rising edge of the WDOGCLK signal;
- Monitor interrupt, 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 Register Definition

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

Table 6-1 Definition of Watch Dog Register

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

Register Name	Address Offset	Type	Width	Initial Value	Description
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 Definition

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

Table 6-2 Initialization of Watch Dog

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 Function Library

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

Table 6-3 Watch Dog Usage

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 provides Watch Dog reference design for Keil and GNU software environment:

- MCU_RefDesign\Keil_RefDesign\watchdog
- MCU_RefDesign\GNU_RefDesign\m1_watchdog

7 GPIO

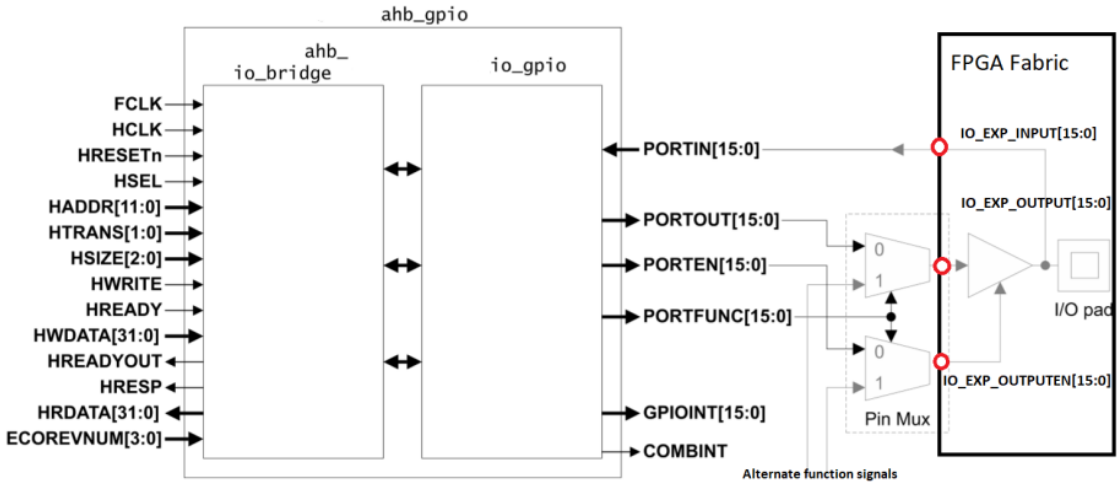
7.1 Features

Gowin_EMPU_M1 contains a GPIO module with a 16-bit input and output interface accessed by one AHB1 bus:

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

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

Figure 7-1 GPIO Block



7.2 Register Definition

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

Table 7-1 GPIO Register Definition

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 delya of two cycle
DATAOUT	0x0004	RW	16	0x0000	[15:0] Data output register value

Register Name	Address Offset	Type	Width	Initial Value	Description
					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] Alernative 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] Alernative 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 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

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 effectx Read back IRQ status register
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 Definition

The GPIO initialization definition 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 Usage of Function Library

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

Table 7-3 Usage of GPIO Function Library

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
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

7.5 Reference Design

Gowin provides GPIO reference design for Keil and GNU software environment:

- MCU_RefDesign\Keil_RefDesign\led
- MCU_RefDesign\GNU_RefDesign\m1_led

8 Operation System

Gowin_EMPU_M1 supports the operation system of uC/OS-III and FreeRTOS.

8.1 uC/OS-III

8.1.1 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 communication between tasks, etc.
- uC/OS-III also provides features that are not available for the other real-time cores. For example, it can measure operation performance during runtime and send signals or messages to tasks directly. Tasks can also wait for multiple semaphores and message queues simultaneously.
- Gowin provides the uC/OS-III reference design that has been transplanted into Gowin_EMPU_M1 successfully. Users can also download from the uC/OS-III website: http://www.micrium.com.

8.1.2 Operation System Version

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

uC/OS-III operation system includes:

- uC-CPU
- uC-LIB
- UCOS_BSP
- uCOS_CONFIG
- uCOS-III
  - Ports
  - Source

8.1.3 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.

8.1.4 Reference Design

Gowin provides uC/OS-III reference design for Keil and GNU software environment:

- MCU_RefDesign\Keil_RefDesign\ucos_iii
- MCU_RefDesign\GNU_RefDesign\m1_ucos_iii

8.2 FreeRTOS

8.2.1 Features

- FreeRTOS is a mini real-time operation system.
- FreeRTOS is a lightweight operation 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 operation system. It has features of open source, portability, reducibility, and flexible scheduling policy.
- Gowin provides the FreeRTOS reference design that has been transplanted into Gowin_EMPU_M1 successfully. Users can also download from the FreeRTOS website: http://www.FreeRTOS.org.

8.2.2 Operation System Version

Gowin_EMPU_M1 reference design uses FreeRTOS V3.03.00 version.

8.2.3 Operation System Configuration

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

8.2.4 Reference Design

Gowin provides FreeRTOS reference design for Keil and GNU software environment:

- MCU_RefDesign\Keil_RefDesign\free_rtos
- MCU_RefDesign\GNU_RefDesign\m1_free_rtos


GOWIN
PROGRAMMING FOR FUTURE