GW1NS-2C MCU

Software Programming Reference Manual

RN516-1.1E, 2018-11-30

Disclaimer

Revision

Date	Version	Description
2018/08/30	1.0E	Initial version.
2018/11/30	1.1E	● Optimize the software programming library; ● Support uC/OS-III and FreeRTOS operating systems.

Contents

List of Figures

List of Tables

1 Software programming library

Gowin provides GW1NS-2C MCU software programming library:

Gowin GW1NS-2C_MCU_PACK\ Gowin_GW1NS-2C_MCU_Src\c_lib。

The software programming library includes an MCU firmware library and an MCU operating system library.

1.1 MCU firmware library

The MCU firmware library is shown in Table 1-1.

Table 1-1 GW1NS-2C MCU firmware library

File	Description
startup_gw1ns2c.s	Boot loader
core_cm3.c	ARM Cortex-M3 inner core definition
gw1ns2c.h	Register definition and address mapping
system_gw1ns2c.c	System initialization and system clock definition
gw1ns2c_flash.ld	Flash link script
gw1ns2c_adc.c	ADC driving function definition
gw1ns2c_gpio.c	GPIO driving function definition
gw1ns2c_timer.c	Timer driving function definition
gw1ns2c_wdog.c	WatchDog driving function definition
gw1ns2c_spi.c	SPI driving function definition
gw1ns2c_misc.c	Interrupt Management and SysTick
gw1ns2c_it.c	Interrupt definition

1.2 MCU operating system library

MCU supports the uC/OS-III and FreeRTOS operating systems.

MCU operating system library includes the following two operating systems:

- uC/OS-III
- FreeRTOS

2Storage System

2.1 Standard Peripheral Memory Map

The memory mapped address of the MCU standard peripheral is shown in Table 2-1.

Table 2-1 GW1NS-2C standard peripheral memory mapping

Standard peripheral	Types	Address mapping	Description
Flash		0x00000000	128K bytes of Flash
Sram		0x20000000	8K byte of Block Ram
TIMER0	TIMER_TypeDef	0x40000000	Timer 0
TIMER1	TIMER_TypeDef	0x40001000	Timer 1
UART0	UART_TypeDef	0x40004000	Universal Asynchronous Receiver 0
UART1	UART_TypeDef	0x40005000	Universal Asynchronous Receiver 1
WatchDog	WDOG_TypeDef	0x40008000	WatchDog
GPIO0	GPIO_TypeDef	0x40010000	General input and output port
SYSCON	SYSCON_TypeDef	0x4001F000	System control
ADC	ADC_TypeDef	0x40002100	Analog/digital conversion
SPI	IMONITOR_TypeDef	0x40002200	Serial peripheral interface
Interrupt Monitor	SPI_TypeDef	0x40002500	Interrupt monitor

2.2 Inner core system memory map

The memory mapping of the MCU inner core system is shown in Table 2-2.

Table 2-2 System Control Memory Mapping

System control	Types	Address mapping	Description
ITM	ITM_Type	0xE0000000	ITM configuration struct
DWT	DWT_Type	0xE0001000	DWT configuration struct
CoreDebug	CoreDebug_Type	0xE000EDF0	Core Debug configuration struct
ETM	ETM_Type	0xE0041000	ETM configuration struct
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
TPIU	TPIU_Type	0xE0040000	TPIU configuration struct

3 Interrupt processing

The nested vectored interrupt controller includes the following features:

- Supports for up to 26 low latency interrupts
- Provides 2 interrupt processing signals available to the user (USER_INT0 and USER_INT1)
- Supports 0-7 programmable priority
- Low latency interrupt and exception handling
- Interrupt signal edge and pulse detection
- Interrupt priority dynamic adjustment

MCU interrupt controller is shown in Table 3-1.

Table 3-1 GW1NS-2C Interrupt Controller

Address	Interrupt	Number	Description
0x00000000	__StackTop		Top of Stack
0x00000004	Reset_Handler		Reset Handler
0x00000008	NMI_Handler		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
0x00000034	0		Reserved
0x00000038	PendSV_Handler	-2	PendSV Handler

Address	Interrupt	Number	Description
0x0000003C	SysTick_Handler	-1	SysTick Handler
0x00000040	UART0_Handler	0	16+ 0: UART 0 RX and TX Handler
0x00000044	Spare1_Handler	1	16+ 1: Not Used
0x00000048	UART1_Handler	2	16+ 2: UART 1 RX and TX Handler
0x0000004C	Spare3_Handler	3	16+ 3: Not Used
0x00000050	Spare4_Handler	4	16+ 4: Not Used
0x00000054	0		16+ 5: Reserved
0x00000058	PORT0_COMB_Handler	6	16+ 6: GPIO Port 0 Combined Handler
0x0000005C	Spare7_Handler	7	16+ 7: Not Used
0x00000060	TIMER0_Handler	8	16+ 8: TIMER 0 handler
0x00000064	TIMER1_Handler	9	16+ 9: TIMER 1 handler
0x00000068	0		16+10: Reserved
0x0000006C	Spare11_Handler	11	16+11: Not Used
0x00000070	UARTOVF_Handler	12	16+12: UART 0,1 Overflow Handler
0x00000074	USER_INT0	13	16+13: USER_INT0 (default System Error Handler for FlashERR)
0x00000078	USER_INT1	14	16+14: USER_INT1 (default Embedded Flash Handler for FLASHINT)
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
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

Address	Interrupt	Number	Description
			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 Transceiver

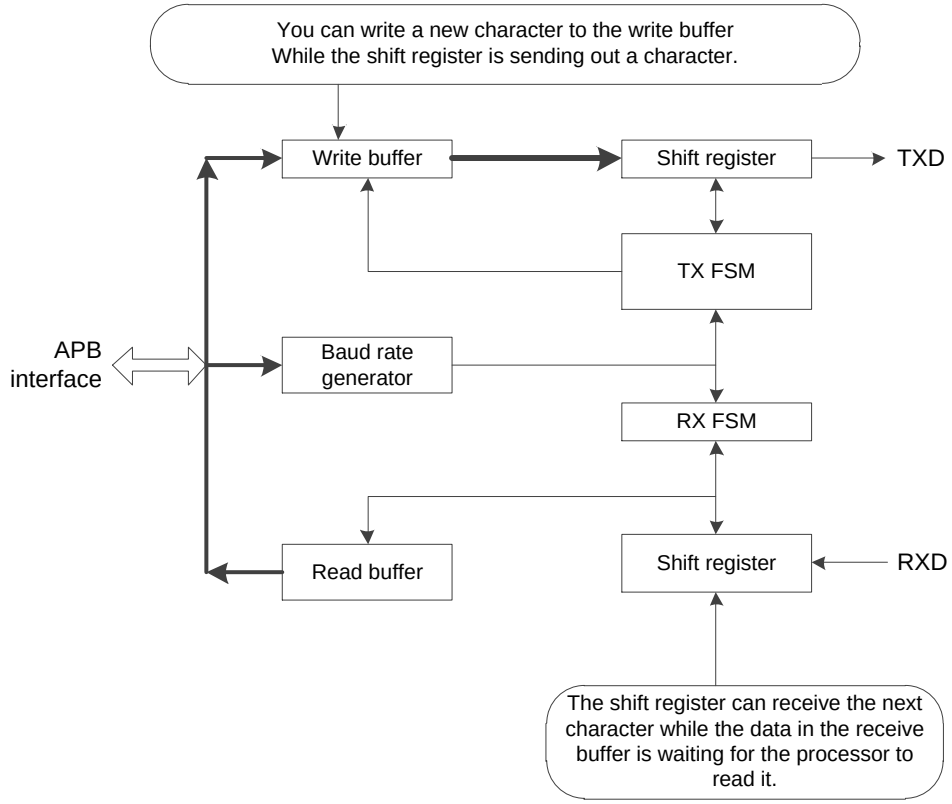
4.1 Features

MCU contains two Universal Asynchronous Receiver UARTs accessed via the APB1 bus:

- The maximum baud rate is 921.6Kbit/s
- No parity check bits
- 8 bits of data bits
- 1 bit of stop bit

UART Buffering is shown in Figure 4-1.

Figure 4-1 UART Buffering



UART supports HSTM（High Speed Test Mode）. 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.

When the user enables UART, the baud rate divider register must be set. 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 structure

The UART register structure definition is shown in Table 4-1.

Table 4-1 UART Register

Register name	Address offset	Types	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

Register name	Address offset	Types	Width	Initial value	Description
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 structure

The UART initialization structure is shown in Table 4-2.

Table 4-2 UART Initialization Definition

Name	Types	Numerical 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 Instructions

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

Table 4-3 How to use UART

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

Name	Description
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
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:

Gowin GW1NS-2C MCU PACK\Gowin_GW1NS-2C_MCU_RefDesign\MCU_RefDesign\Keil\uart

Gowin GW1NS-2C MCU PACK\Gowin_GW1NS-2C_MCU_RefDesign\MCU_RefDesign\GNU\uart

5 Timer

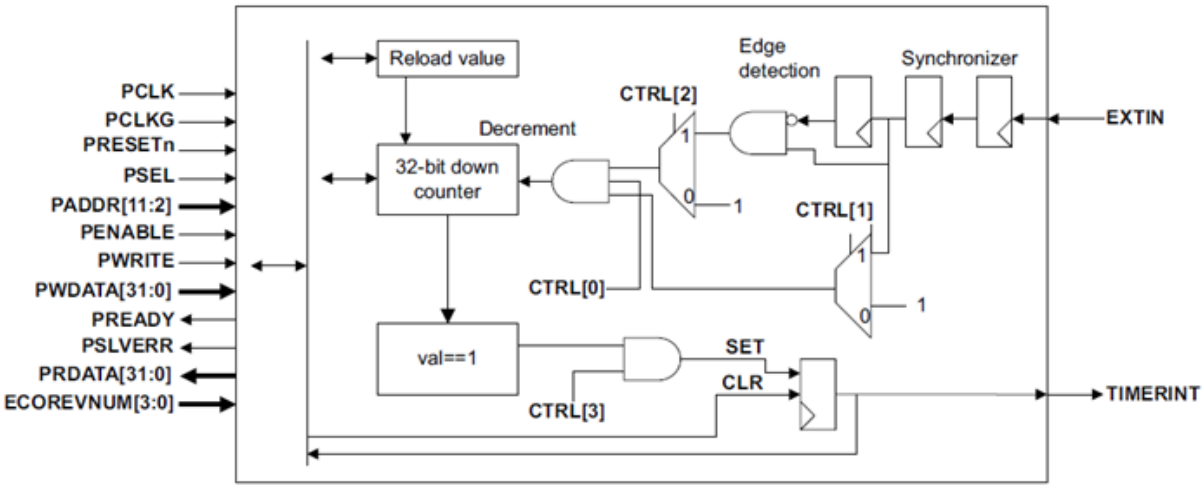
5.1 Features

MCU contains two synchronous standard timers accessed via the APB1 bus:

- 32-bit counter
- Can generate interrupt request signal
- The clock can be enabled by using the external input signal EXTIN
- TIMER0: EXTIN connects GPIO[1]
- TIMER1: EXTIN connects GPIO[6]

The TIMER structure is shown in Figure 5-1.

Figure 5-1 TIMER



5.2 Register structure

The structure of the TIMER register is shown in Table 5-1.

Table 5-1 TIMER register structure

Register name	Address offset	Types	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 structure

The TIMER initialization definition is shown in Table 5-2.

Table 5-2 TIMER initialization structure

Name	Types	Numerical 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 Instructions

The usage of the TIMER function library is shown in Table 5-3.

Table 5-3 How to use TIMER

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

Name	Description
TIMER_SetValue	Sets TIMERx current value
TIMER_EnableIRQ	Enable TIMERx interrupt request
TIMER_DisableIRQ	Disable TIMERx interrupt request

5.5 Reference design

Gowin provides the Timer reference design:

Gowin GW1NS-2C MCU PACK\Gowin_GW1NS-2C_MCU_RefDesign\MCU_RefDesign\GNU\timer

6 WatchDog

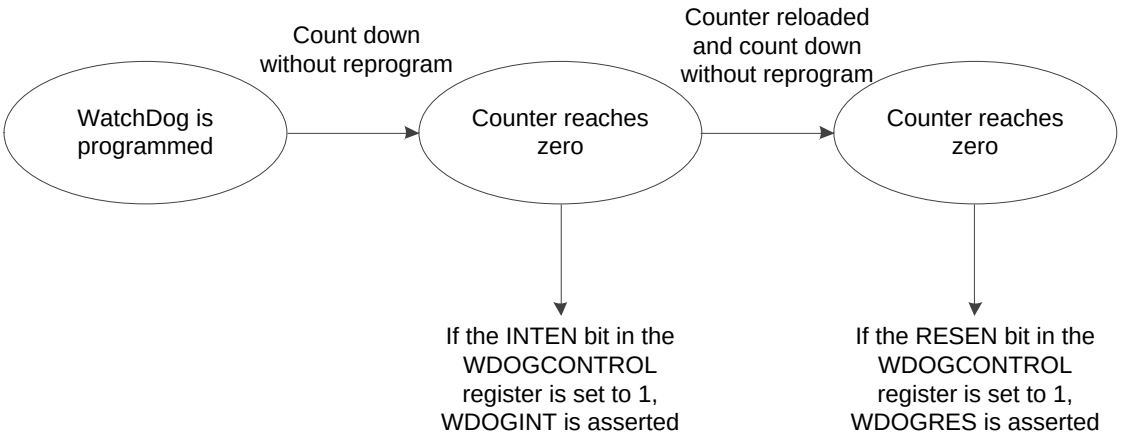
6.1 Features

MCU contains a WatchDog accessed via the APB1 bus:

- Based on the 32-bit down-counter initialized by the LOAD register
- Generates an interrupt request
- When the clock is enabled, the counter is decremented by the rising edge of the WDOGCLK signal.
- Monitor interrupts, when the counter is decremented to 0, a reset request is generated and the counter is stopped.
- Responds to software reset caused by software crash, provide software recovery method

The WatchDog operation is shown in Figure 6-1.

Figure 6-1 WatchDog Operation



6.2 Register structure

The structure of the WatchDog register is shown in Table 6-2.

Table 6-1 WatchDog register structure

Register name	Address offset	Types	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 structure

The WatchDog initialization structure is shown in Table 6-2.

Table 6-2 WatchDog initialization structure

Name	Types	Numerical 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 How to use the function library

The usage of the WatchDog function library is shown in Table 6-3.

Table 6-3 How to use WatchDog

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 the WatchDog reference design: Gowin GW1NS-2C MCU PACK\Gowin_GW1NS-2C_MCU_RefDesign\MCU_RefDesign\GNU\watchdog

7 General input and output

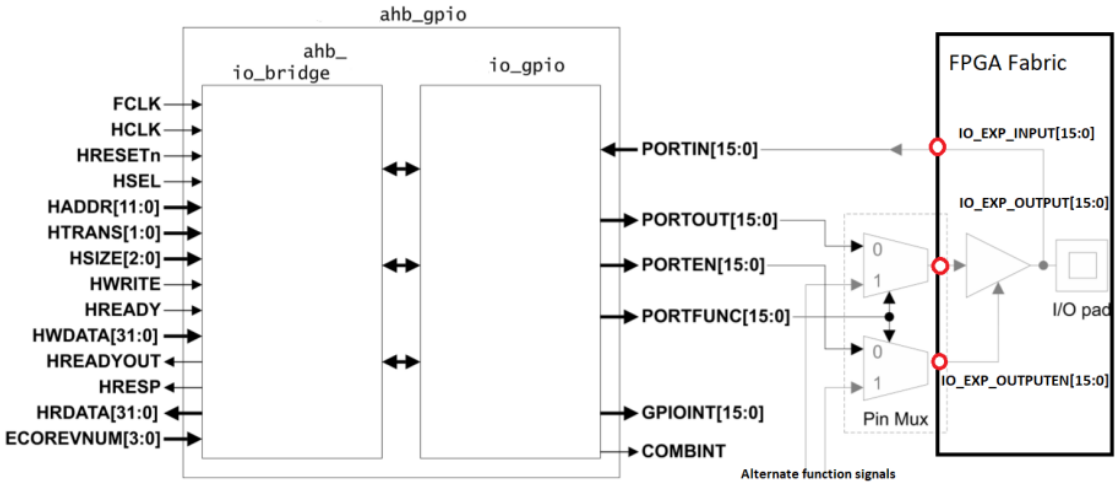
7.1 Features

MCU contains a GPIO module with a 16-bit input and output interface accessed via the AHB bus:

- Connects to the FPGA architecture;
- Each IO pin can generate an interrupt;
- Supports bit mask;
- Pin multiplexing function.

The GPIO architecture is shown in Figure 7-1.

Figure 7-1 GPIO Block



7.2 Register structure

The GPIO register structure is shown in Table 7-1.

Table 7-1 GPIO Register Structure

Register Name	Address Offset	Types	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 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

Register Name	Address Offset	Types	Width	Initial Value	Description
					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 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

Register Name	Address Offset	Types	Width	Initial Value	Description
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 shown in Table 7-2.

Table 7-2 GPIO initialization structure

Name	Types	Numerical 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 How to use the function library

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

Table 7-3 How to use GPIO

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:

Gowin GW1NS-2C MCU PACK\Gowin_GW1NS-2C_MCU_RefDesign\MCU_RefDesign\Keil\lcd

Gowin GW1NS-2C MCU PACK\Gowin_GW1NS-2C_MCU_RefDesign\MCU_RefDesign\Keil\led

Gowin GW1NS-2C MCU PACK\Gowin_GW1NS-2C_MCU_RefDesign\MCU_RefDesign\GNU\lcd

Gowin GW1NS-2C MCU PACK\Gowin_GW1NS-2C_MCU_RefDesign\MCU_RefDesign\GNU\led

8 Analog/digital conversion

8.1 Register definition

The ADC register definitions are shown in Table 8-1.

Table 8-1 ADC Register Structure

Register name	Address offset	Types	Width	Initial value	Description
DATA	0x00	RO	12	0x000	[11:0] conversion data
STATUS	0x04	RW	2	0x0	[1] Start of conversion status [0] End of conversion status
CTRL	0x08	RW	6	0x00	[5] AD conversion mode [4] AD conversion starting [2:0] Channel

8.2 Initialization definition

The ADC initialization definition is shown in Table 8-2.

Table 8-2 ADC initialization structure

Name	Types	Numerical value	Description
ADC_Mode	uint32_t	ADC_MODE_CONT ADC_MODE_SINGLE	ADC continuous or single conversion mode
ADC_Status	uint32_t	ADC_STATUS_ON ADC_STATUS_OFF	Start or stop conversion
ADC_Chsel	uint32_t	ADC_CHSEL_0 ADC_CHSEL_1	ADC channel 0-7

Name	Types	Numerical value	Description
		ADC_CHSEL_2 ADC_CHSEL_3 ADC_CHSEL_4 ADC_CHSEL_5 ADC_CHSEL_6 ADC_CHSEL_7	

8.3 Instructions

How to use ADC is shown in Table 8-3.

Table 8-3 How to use ADC

Name	Description
ADC_Init	Initializes ADC
ADC_SetMode	Sets ADC conversion mode
ADC_GetMode	Returns ADC conversion mode
ADC_SetPowerStatus	Sets ADC running status
ADC_GetPowerStatus	Returns ADC running status
ADC_SetChannel	Sets ADC conversion channel
ADC_GetChannel	Returns ADC conversion channel
ADC_GetEocStatus	Returns stopping status
ADC_GetSocStatus	Returns starting status
ADC_ReadData	Returns ADC conversion data

9 Serial peripheral interface

9.1 Register definition

The SPI register definition is shown in Table 9-1.

Table 9-1 SPI Register Structure

Register name	Address offset	Types	Width	Initial value	Description
RDATA	0x00	RO	8	0x00	Read data register
WDATA	0x04	WO	8	0x00	Write data register
STATUS	0x08	RW	8	0x00	[7] 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	8	0x00	Unused selected slave address
CTRL	0x10	RW	5	0x00	[4:3] Clock selected [2] Polarity [1] Phase [0] Direction

9.2 Initialization structure

The SPI initialization structure is shown in Table 9-2.

Table 9-2 SPI Initialization Structure

Name	Types	Numerical 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

9.3 How to use the function library

The usage of the SPI function library is shown in Table 9-3.

Table 9-3 How to use SPI

Name	Description
SPI_Init	Initializes SPI
SPI_SetDirection	Sets direction
SPI_ClrDirection	Clears direction
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	Clear receive overrun error status
SPI_ClrErrStatus	Clears error status
SPI_WriteData	Writes data
SPI_ReadData	Reads data

10 Interrupt monitor

10.1 Register definition

The Interrupt Monitor register definition is shown in Table 10-1.

Table 10-1 Interrupt Monitor Register Definitions

Register name	Address offset	Types	Width	Initial value	Description
INTSTATUS	0x00	RW	21	0x000000	[20] WatchDog interrupt status [19] UART1 interrupt status [18] UART0 interrupt status [17] TIMER1 interrupt status [16] TIMER0 interrupt status [15] GPIO0 Pin15 interrupt status [14] GPIO0 Pin14 interrupt status [13] GPIO0 Pin13 interrupt status [12] GPIO0 Pin12 interrupt status [11] GPIO0 Pin11 interrupt status [10] GPIO0 Pin10 interrupt status [9] GPIO0 Pin9 interrupt status [8] GPIO0 Pin8 interrupt status [7] GPIO0 Pin7 interrupt status [6] GPIO0 Pin6 interrupt status [5] GPIO0 Pin5 interrupt status [4] GPIO0 Pin4 interrupt status [3] GPIO0 Pin3 interrupt status [2] GPIO0 Pin2 interrupt status [1] GPIO0 Pin1 interrupt status [0] GPIO0 Pin0 interrupt status

10.2 How to use the function library

The usage of the Interrupt Monitor function library is shown in Table 10-2.

Table 10-2 How to use Interrupt Monitor function library

Name	Description
IMONITOR_Init	Initializes IMONITOR
IMONITOR_SetGPIO0Pin0IntStatus	Sets GPIO0 Pin0 interrupt status
IMONITOR_GetGPIO0Pin0IntStatus	Returns GPIO0 Pin0 interrupt status
IMONITOR_ClrGPIO0Pin0IntStatus	Clears GPIO0 Pin0 interrupt status
IMONITOR_SetGPIO0Pin1IntStatus	Sets GPIO0 Pin1 interrupt status
IMONITOR_GetGPIO0Pin1IntStatus	Returns GPIO0 Pin1 interrupt status
IMONITOR_ClrGPIO0Pin1IntStatus	Clears GPIO0 Pin1 interrupt status
IMONITOR_SetGPIO0Pin2IntStatus	Sets GPIO0 Pin2 interrupt status
IMONITOR_GetGPIO0Pin2IntStatus	Returns GPIO0 Pin2 interrupt status
IMONITOR_ClrGPIO0Pin2IntStatus	Clears GPIO0 Pin2 interrupt status
IMONITOR_SetGPIO0Pin3IntStatus	Sets GPIO0 Pin3 interrupt status
IMONITOR_GetGPIO0Pin3IntStatus	Returns GPIO0 Pin3 interrupt status
IMONITOR_ClrGPIO0Pin3IntStatus	Clears GPIO0 Pin3 interrupt status
IMONITOR_SetGPIO0Pin4IntStatus	Sets GPIO0 Pin4 interrupt status
IMONITOR_GetGPIO0Pin4IntStatus	Returns GPIO0 Pin4 interrupt status
IMONITOR_ClrGPIO0Pin4IntStatus	Clears GPIO0 Pin4 interrupt status
IMONITOR_SetGPIO0Pin5IntStatus	Sets GPIO0 Pin5 interrupt status
IMONITOR_GetGPIO0Pin5IntStatus	Returns GPIO0 Pin5 interrupt status
IMONITOR_ClrGPIO0Pin5IntStatus	Clears GPIO0 Pin5 interrupt status
IMONITOR_SetGPIO0Pin6IntStatus	Sets GPIO0 Pin6 interrupt status
IMONITOR_GetGPIO0Pin6IntStatus	Returns GPIO0 Pin6 interrupt status

Name	Description
IMONITOR_ClrGPIO0Pin6IntStatus	Clears GPIO0 Pin6 interrupt status
IMONITOR_SetGPIO0Pin7IntStatus	Sets GPIO0 Pin7 interrupt status
IMONITOR_GetGPIO0Pin7IntStatus	Returns GPIO0 Pin7 interrupt status
IMONITOR_ClrGPIO0Pin7IntStatus	Clears GPIO0 Pin7 interrupt status
IMONITOR_SetGPIO0Pin8IntStatus	Sets GPIO0 Pin8 interrupt status
IMONITOR_GetGPIO0Pin8IntStatus	Returns GPIO0 Pin8 interrupt status
IMONITOR_ClrGPIO0Pin8IntStatus	Clears GPIO0 Pin8 interrupt status
IMONITOR_SetGPIO0Pin9IntStatus	Sets GPIO0 Pin9 interrupt status
IMONITOR_GetGPIO0Pin9IntStatus	Returns GPIO0 Pin9 interrupt status
IMONITOR_ClrGPIO0Pin9IntStatus	Clears GPIO0 Pin9 interrupt status
IMONITOR_SetGPIO0Pin10IntStatus	Sets GPIO0 Pin10 interrupt status
IMONITOR_GetGPIO0Pin10IntStatus	Returns GPIO0 Pin10 interrupt status
IMONITOR_ClrGPIO0Pin10IntStatus	Clears GPIO0 Pin10 interrupt status
IMONITOR_SetGPIO0Pin11IntStatus	Sets GPIO0 Pin11 interrupt status
IMONITOR_GetGPIO0Pin11IntStatus	Returns GPIO0 Pin11 interrupt status
IMONITOR_ClrGPIO0Pin11IntStatus	Clears GPIO0 Pin11 interrupt status
IMONITOR_SetGPIO0Pin12IntStatus	Sets GPIO0 Pin12 interrupt status
IMONITOR_GetGPIO0Pin12IntStatus	Returns GPIO0 Pin12 interrupt status
IMONITOR_ClrGPIO0Pin12IntStatus	Clears GPIO0 Pin12 interrupt status
IMONITOR_SetGPIO0Pin13IntStatus	Sets GPIO0 Pin13 interrupt status
IMONITOR_GetGPIO0Pin13IntStatus	Returns GPIO0 Pin13 interrupt status
IMONITOR_ClrGPIO0Pin13IntStatus	Clears GPIO0 Pin13 interrupt status
IMONITOR_SetGPIO0Pin14IntStatus	Sets GPIO0 Pin14 interrupt status
IMONITOR_GetGPIO0Pin14IntStatus	Returns GPIO0 Pin14 interrupt status

Name	Description
IMONITOR_ClrGPIO0Pin14IntStatus	Clears GPIO0 Pin14 interrupt status
IMONITOR_SetGPIO0Pin15IntStatus	Sets GPIO0 Pin15 interrupt status
IMONITOR_GetGPIO0Pin15IntStatus	Returns GPIO0 Pin15 interrupt status
IMONITOR_ClrGPIO0Pin15IntStatus	Clears GPIO0 Pin15 interrupt status
IMONITOR_SetTIMER0IntStatus	Sets TIMER0 interrupt status
IMONITOR_GetTIMER0IntStatus	Returns TIMER0 interrupt status
IMONITOR_ClrTIMER0IntStatus	Clears TIMER0 interrupt status
IMONITOR_SetTIMER1IntStatus	Sets TIMER0 interrupt status
IMONITOR_GetTIMER1IntStatus	Returns TIMER0 interrupt status
IMONITOR_ClrTIMER1IntStatus	Clears TIMER0 interrupt status
IMONITOR_SetUART0IntStatus	Sets UART0 interrupt status
IMONITOR_GetUART0IntStatus	Returns UART0 interrupt status
IMONITOR_ClrUART0IntStatus	Clears UART0 interrupt status
IMONITOR_SetUART1IntStatus	Sets UART1 interrupt status
IMONITOR_GetUART1IntStatus	Returns UART1 interrupt status
IMONITOR_ClrUART1IntStatus	Clears UART1 interrupt status
IMONITOR_SetWDOGIntStatus	Sets WatchDog interrupt status
IMONITOR_GetWDOGIntStatus	Returns WatchDog interrupt status
IMONITOR_ClrWDOGIntStatus	Clears WatchDog interrupt status

11 System controller

11.1 Register definition

The SYSCON register definition is shown in Table 11-1.

Table 11-1 SYSCON Register Definition

Register name	Address offset	Types	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

11.2 How to use the function library

How to use the SYSCON library is shown in Table 11-2.

Table 11-2 How to use the SYSCON function library

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
SYSCON_GetRstinfoWdogresetreq	Returns RSTINFO SYSRESETREQ
SYSCON_GetRstinfoLockreset	Returns RSTINFO SYSRESETREQ

12 Operating system

MCU supports uC/OS-III and FreeRTOS operating systems.

12.1 uC/OS-III

12.1.1 Features

- UC/OS-III is an extensible, curable, preemptive real-time inner core with unlimited number of tasks managed
- UC/OS-III is a third-generation inner core that provides the functionality expected of modern real-time inner cores, including resource management, synchronization, inter-task communication, etc.
- UC/OS-III provides many features that are not available in other real-time inner cores, such as measuring run performance at runtime, sending signals or messages directly to tasks, and tasks can simultaneously wait for multiple semaphores and message queues.
- Gowin provides the uC/OS-III operating system library that has successfully transplanted Gowin development environment, or downloaded from uC/OS-III official website http://www.micrium.com

12.1.2 Operating system library

The uC/OS-III version provided by the MCU software programming library is V3.03.00.

The uC/OS-III operating system library includes:

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

12.1.3 Operating system configuration

Users can configure uC/OS-III by modifying UCOSIII_CONFIG\os_cfg.h and os_cfg_app.h.

12.1.4 Reference design

Gowin provides a reference design for uC/OS-III based on GNU MCU Eclipse and ARM KEIL software:

Gowin GW1NS-2C MCU PACK\Gowin_GW1NS-2C_MCU_RefDesign\MCU_RefDesign\Keil\ucos_iii

Gowin GW1NS-2C MCU PACK\Gowin_GW1NS-2C_MCU_RefDesign\MCU_RefDesign\GNU\ucos_iii

12.2 FreeRTOS

12.2.1 Features

- FreeRTOS is a mini real-time operating system
- FreeRTOS as a lightweight operating system, including task management, time management, semaphores, message queues, memory management, logging functions, software timers, coroutines, etc., can basically meet the needs of smaller systems
- FreeRTOS operating system is a completely free operating system with open source, portable, scalable, and flexible scheduling strategies.
- Gowin provides the FreeRTOS operating system library that has successfully transplanted Gowin development environment, or downloaded from the FreeRTOS official website http://www.FreeRTOS.org

12.2.2 Operating system library

The version of FreeRTOS provided by the MCU software programming library is V9.0.0.

12.2.3 Operating system configuration

Users can configure FreeRTOS by modifying include\FreeRTOSConfig.h.

12.2.4 Reference design

Gowin offers FreeRTOS reference designs based on GNU MCU Eclipse and ARM KEIL software:

Gowin GW1NS-2C MCU PACK\Gowin_GW1NS-2C_MCU_RefDesign\MCU_RefDesign\Keil\free_rtos

Gowin GW1NS-2C MCU PACK\Gowin_GW1NS-2C_MCU_RefDesign\MCU_RefDesign\GNU\free_rtos


GOWIN
PROGRAMMING FOR FUTURE