Gowin OC8051 MCU User Guide

GOWIN Semiconductor Corp.

Embedded System RnD

2024-09-26

Revision History

Date	Version	Description
2024/09/26	1.0	Initial version published.

Content

1 About This Manual

1.1 Purpose

This manual primarily describes the functional overview of the OC8051 microcontroller, signal description, hardware parameter configuration, software register configuration, usage of the IDE software, reference design, and the development process. It aims to help users quickly master the software and hardware development methods of the OC8051 microcontroller, thereby saving development time and improving efficiency.

1.2 Terminology and Abbreviation

The relevant terminology, abbreviations, and their definitions in this manual are shown in Table 1-1.

Table 1-1 Terminology and Abbreviations

Terminology and abbreviations	Full name	Description
IDE	Integrated Development Environment	Integrated development environment.
UART	Universal Synchronous and Asynchronous Receiver/Transmitter	Universal synchronous and asynchronous receiver/transmitter.
PSW	Program Status Word	Program status word.
IE	Interrupt Enable	Enable interrupt.
IP	Interrupt Priority	Interrupt priority.
DPL	Data Pointer Low	Data pointer low bits.
DPH	Data Pointer High	Data pointer high bits.
TMOD	Timer Mode	Timer mode.
TCON	Timer Control	Control timer.
SCON	Serial Control	Control serial interface.
SBUF	Serial Buffer	Serial interface buffer.
PCON	Power Control	Control power.
SP	Stack Pointer	Stack pointer.
RCAP2L	Timer2 Reload and Capture Low	Timer2 reload and capture low bits.
RCAP2H	Timer2 Reload and Capture High	Timer2 reload and capture high bits.

1.3 Support and Feedback

Gowin Semiconductor provides customers with comprehensive technical support. If you have any questions, comments, or suggestions, please feel free to contact us directly using the information provided below.

Website：www.gowinsemi.com

E-mail：support@gowinsemi.com

2 Functional Description

2.1 Overview

The OC8051 microcontroller is a member of the MCS-51 series, originally designed by Intel in the 1980s. Since its launch, the 8051 has gained widespread popularity and continues to have significant market applications in low-end embedded systems.

The basic components of the 8051 microcontroller include multiple on-chip peripherals, such as I/O ports and timers/counters, as well as up to 256 bytes of on-chip data memory and up to 64KB of on-chip program memory.

As a soft-core microcontroller, the OC8051 overcomes the hardware limitations of the 8051 chip, allowing for the expansion of on-chip ROM and RAM space based on signal width to run larger program code. It also enables the separation of the P3 interface from external interrupt inputs and timer/counter inputs, allowing for independent operation and freeing up multiplexed pins.

2.2 Feature

The features of the OC8051 microcontroller include:

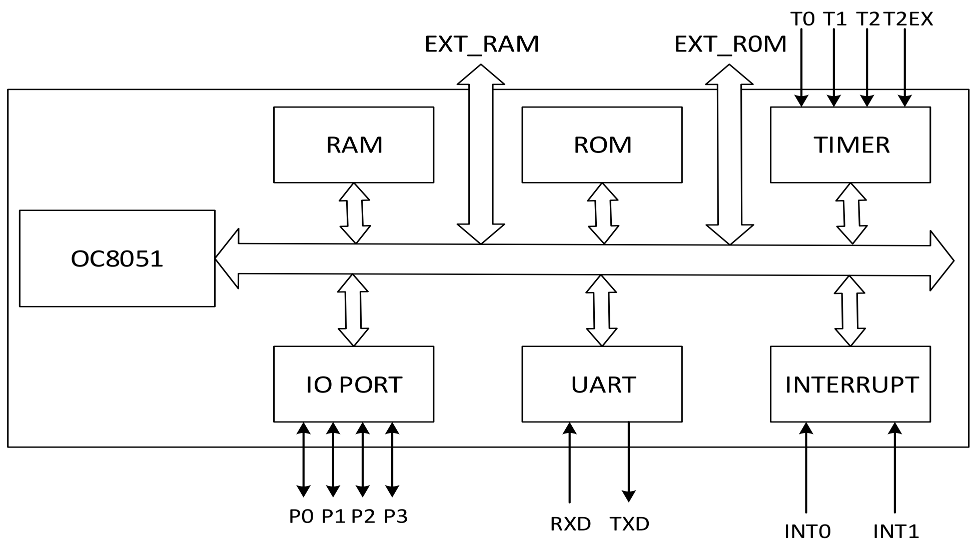
- An 8-bit microcontroller optimized for control applications
- Extensive Boolean processing (single-bit logic) capabilities
- 64KB external program memory addressing space
- 64KB external data memory addressing space
- Up to 64KB of on-chip program ROM
- Up to 256 bytes of on-chip data RAM
- Two sets of buses for connecting external memory
- 32 bidirectional and individually addressable I/O lines

- Three 16-bit timers/counters
- One UART interface
- Six interrupt sources with two priority levels

2.3 Block Diagram

The structural block diagram of the OC8051 microcontroller is shown in Figure 2-1.

Figure 2-1 Block Diagram



3 Signal Description

The signal description of the OC8051 microcontroller is shown in Table 3-1.

Table 3-1 Signal Description

Signal name	I/O	Width	Description	Belongs
wb_clk_i	in	1	System clock signal.	-
wb_rst_i	in	1	System reset signal.	-
wbi_adr_o	out	16	External ROM address input signal.	External ROM access
wbi_dat_i	in	32	External ROM data input signal.	
wbi_stb_o	out	1	External ROM byte strobe output signal.	
wbi_ack_i	in	1	External ROM response input signal.	
wbi_cyc_o	out	1	External ROM bus cycle signal.	
wbi_err_i	in	1	External ROM transmission error signal.	
ea_in	in	1	External ROM access enable signal.	
wbd_dat_i	in	8	External RAM data input signal.	External RAM access
wbd_dat_o	out	8	External RAM data output signal.	
wbd_adr_o	out	16	External RAM address output signal.	
wbd_we_o	out	1	External RAM data write/read enable signal.	
wbd_ack_i	in	1	External RAM response input signal.	
wbd_stb_o	out	1	External RAM byte strobe output signal.	
wbd_cyc_o	out	1	External RAM bus cycle output signal.	
wbd_err_i	in	1	External RAM transmission error signal.	
int0_i	in	1	External interrupt 0 signal.	External interrupt
int1_i	in	1	External interrupt 1 signal.	
p0_i	in	8	P0 port input signal.	I/O PORT
p0_o	out	8	P0 port output signal.	
p1_i	in	8	P1 port input signal.	
p1_o	out	8	P1 port output signal.	
p2_i	in	8	P2 port input signal.	
p2_o	out	8	P2 port output signal.	
p3_i	in	8	P3 port input signal.	
p3_o	out	8	P3 port output signal.	
rxd_i	in	1	UART receive signal.	UART
txd_o	out	1	UART send signal.	
t0_i	in	1	Timer/Counter 0 input signal.	TIMER
t1_i	in	1	Timer/Counter 1 input signal.	
t2_i	in	1	Timer/Counter 2 input signal.	
t2ex_i	in	1	Timer/Counter 2 capture/reload trigger control signal.	
scanb_rst	in	1	BIST reset input signal.	BIST
scanb_clk	in	1	BIST clock input signal.	

scanb_si	in	1	BIST bit input signal.	
scanb_so	out	1	BIST bit output signal.	
scanb_en	in	1	BIST bit enable signal.	

4 Hardware Parameter Configuration

The hardware parameter configuration of OC8051 microcontroller is shown in Table 4-1.

It is located in …\oc8051\src\oc8051_defines.v.

Table 4-1 Hardware Parameter Configuration

Parameter option	Description
INT_ROM_SIZE	On-chip program memory size 65536: 64KB 32768: 32KB 16384: 16KB 8192: 8KB 4096: 4KB 2018: 2KB 1024: 1KB Default: 4096.
INT_RAM_SIZE	On-chip data memory size 8: 256B 7: 128B 6: 64B 5: 32B 4: 16B Default: 8.
OC8051_CODES_FILE	Software program code file.
OC8051_UART	Enable UART.
OC8051_TC01	Enable Timer/Counter 0 and Timer/Counter 1.
OC8051_TC2	Enable Timer/Counter 2.
OC8051_PORTS	Enable I/O PORTS.
OC8051_PORT0	Enable P0.
OC8051_PORT1	Enable P1.
OC8051_PORT2	Enable P2.
OC8051_PORT3	Enable P3.
OC8051_ROM	Enable on-chip program memory.
OC8051_GENERIC	Enable on-chip data memory.
OC8051_CACHE	Enable Cache with external memory.
OC8051_WB	Enable Wishbone interface.
OC8051_SIMULATION	Define simulation.
OC8051_BIST	Built-in self test.

5 Software Register Configuration

The software registers, addresses, and descriptions of the OC8051 microcontroller are shown in Table 5-1.

Table 5-1 Software Register Configuration

Register	Address	Description
ACC	0xE0	Accumulator register.
B	0xF0	B register: Used to hold the multiplier or divisor during multiplication or division operations.
PSW	0xD0	Program Status Word (PSW): Used to store the operating state of the OC8051 microcontroller.
IE	0xA8	Interrupt enable register.
IP	0xB8	Interrupt priority control register.
P0	0x80	P0 I/O register.
P1	0x90	P1 I/O register.
P2	0xA0	P2 I/O register.
P3	0xB0	P3 I/O register.
DPL	0x82	Low 8 bits of the data pointer (DPTR).
DPH	0x83	High 8 bits of the data pointer (DPTR).
TMOD	0x89	Timer/Counter mode register.
TCON	0x88	Timer/Counter control register.
TL0	0x8A	Low 8-bit data fill register for Timer/Counter 0.
TH0	0x8C	High 8-bit data fill register for Timer/Counter 0.
TL1	0x8B	Low 8-bit data fill register for Timer/Counter 1.
TH1	0x8D	High 8-bit data fill register for Timer/Counter 1.
SCON	0x98	Serial communication control register.
SBUF	0x99	Serial data buffer register.
PCON	0x87	Power control register.
SP	0x81	Stack pointer register.
RCAP2L	0xCA	Low 8-bit reload/capture register for Timer/Counter 2.
RCAP2H	0xCB	High 8-bit reload/capture register for Timer/Counter 2.
T2CON	0xC8	Control register for Timer/Counter 2.
TL2	0xCC	Low 8-bit data fill register for Timer/Counter 2.
TH2	0xCD	High 8-bit data fill register for Timer/Counter 2.

5.1 ACC Register

The ACC register is the most commonly used register in the 8051 microcontroller, also referred to as A. It is frequently used to store operands and results of arithmetic or logical operations.

5.2 B Register

The B register must be used in conjunction with the ACC register during multiplication and division operations.

The MUL AB instruction multiplies the 8-bit unsigned numbers in accumulator A and register B. The low byte of the 16-bit product is stored in A, and the high byte is stored in B.

The DIV AB instruction divides the value in B by the value in A. The integer quotient is stored in A, and the remainder is stored in B.

Additionally, the B register can be used as a general-purpose temporary register.

5.3 PSW Register

The PSW (Program Status Word) register contains several status bits that represent the current state of the microcontroller.

The PSW register is shown in Table 5-2.

Table 5-2 PSW Register

0xD7	0xD6	0xD5	0xD4	0xD3	0xD2	0xD1	0xD0
CY	AC	F0	RS1	RS0	OV		P

The PSW register includes the following status bits:

- CY (Carry Flag): This flag is set to 1 if there is a carry from the most significant bit during an addition, or if there is a borrow during a subtraction. Otherwise, it is 0.
- AC (Auxiliary Carry Flag): This flag is set to 1 if there is a carry from the low nibble (4 bits) to the high nibble during an addition, or if there is a borrow from the low nibble during a subtraction. Otherwise, it is 0.
- F0 (User Flag): A user-defined flag bit.
- RS1, RS0: Register bank selection bits, as shown in Table 5-3.

Table 5-3 RS1、RS0 Bit

RS1	RS0	Currently used register bank	Address
0	0	0 bank	00H-07H
0	1	1 bank	08H-0FH
1	0	2 bank	10H-17H
1	1	3 bank	18H-1FH

- OV (Overflow Flag): This flag indicates whether an overflow has occurred during arithmetic operations.
- P (Parity Flag): This flag is used to indicate the parity of the number of 1s in the ACC register. If the count of 1s is odd, P is set to 1; if the count is even, P is set to 0.

5.4 IE Register

The IE (Interrupt Enable) register is used to control whether each interrupt source is enabled.

The IE register is shown in Table 5-4.

Table 5-4 IE Register

0xAF	0xAE	0xAD	0xAC	0xAB	0xAA	0xA9	0xA8
EA		ET2	ES	ET1	EX1	ET0	EX0

The details of the IE (Interrupt Enable) register are as follows:

- EA (Global Interrupt Enable): When EA is set to 1, all interrupts are enabled; when EA is 0, all interrupts are disabled.
- ET2 (Timer/Counter 2 Overflow Interrupt Enable): When ET2 is set to 1, Timer/Counter 2 interrupt is enabled; when ET2 is 0, Timer/Counter 2 interrupt is disabled.
- ES (Serial Port Interrupt Enable): When ES is set to 1, serial port interrupt is enabled; when ES is 0, serial port interrupt is disabled.

- ET1 (Timer/Counter 1 Overflow Interrupt Enable): When ET1 is set to 1, Timer/Counter 1 interrupt is enabled; when ET1 is 0, Timer/Counter 1 interrupt is disabled.
- EX1 (External Interrupt 1 Enable): When EX1 is set to 1, external interrupt 1 is enabled; when EX1 is 0, external interrupt 1 is disabled.
- ET0 (Timer/Counter 0 Overflow Interrupt Enable): When ET0 is set to 1, Timer/Counter 0 interrupt is enabled; when ET0 is 0, Timer/Counter 0 interrupt is disabled.
- EX0 (External Interrupt 0 Enable): When EX0 is set to 1, external interrupt 0 is enabled; when EX0 is 0, external interrupt 0 is disabled.

5.5 IP Register

The IP (Interrupt Priority) register is used to control the priority of interrupts.

The IP register is shown in Table 5-5.

Table 5-5 IP Register

0xBF	0xBE	0xBD	0xBC	0xBB	0xBA	0xB9	0xB8
		PT2	PS	PT1	PX1	PT0	PX0

The details of the IP (Interrupt Priority) register are as follows:

- PT2 (Timer/Counter 2 Interrupt Priority): When PT2 is set to 1, this interrupt has a higher priority; when PT2 is 0, this interrupt has a lower priority.
- PS (Serial Port Interrupt Priority): When PS is set to 1, this interrupt has a higher priority; when PS is 0, this interrupt has a lower priority.
- PT1 (Timer/Counter 1 Interrupt Priority): When PT1 is set to 1, this interrupt has a higher priority; when PT1 is 0, this interrupt has a lower priority.
- PX1 (External Interrupt 1 Priority): When PX1 is set to 1, this interrupt has a higher priority; when PX1 is 0, this interrupt has a lower priority.

- PT0 (Timer/Counter 0 Interrupt Priority): When PT0 is set to 1, this interrupt has a higher priority; when PT0 is 0, this interrupt has a lower priority.
- PX0 (External Interrupt 0 Priority): When PX0 is set to 1, this interrupt has a higher priority; when PX0 is 0, this interrupt has a lower priority.

5.6 P0, P1, P2, P3 Register

P0, P1, P2, and P3 are the registers for four parallel input/output ports, each containing 8 pins for input and output operations. These ports can be manipulated at the bit level.

5.7 DPL, DPH Register

The Data Pointer Register is a 16-bit dedicated register composed of the low 8 bits (DPL register) and the high 8 bits (DPH register). It can perform data operations on external data RAM within a range of 64KB using either indirect addressing or indexed addressing modes.

5.8 TMOD Register

The TMOD (Timer Mode) register is used to control the operating modes of Timer/Counter 0 and Timer/Counter 1, and it is not bit-addressable.

The TMOD register is shown in Table 5-6.

Table 5-6 TMOD Register

7	6	5	4	3	2	1	0
GATE	C/T	M1	M0	GATE	C/T	M1	M0

The details of the TMOD (Timer Mode) register are as follows:

- GATE: Timer operation switch control bit. When GATE = 1, Timer/Counter 0 or 1 will start working only if the INT0 or INT1 pin is high and the TR0 or TR1 control bit in TCON is set to 1. When GATE = 0, Timer/Counter 0 or 1 will start working as soon as the TR0 or TR1 control bit is set to 1.

- C/T: Function selection bit for Timer/Counter. When C/T = 1, it operates as a counter, counting external pulses from the T0 or T1 input pins. When C/T = 0, it operates as a timer, using the internal system clock to provide timing pulses.
- M0, M1: These are the mode selection bits for the operation modes of Timer/Counter, as shown in Table 5-7.

Table 5-7 M0, M1 Bit

M0	M1	Mode	Description
0	0	0	13-bit Timer/Counter: TH0 or TH1 is fully utilized, while TL0 or TL1 only uses the lower 5 bits.
0	1	1	16-bit Timer/Counter: TH0 or TH1 and TL0 or TL1 are fully utilized.
1	0	2	8-bit Auto-Reload Timer: When it overflows, the value from TH0 or TH1 is automatically reloaded into TL0 or TL1.
1	1	3	Timer/Counter 1: Stops functioning. Timer 0 acts as a dual 8-bit Timer/Counter. TL0 operates as an 8-bit Timer/Counter, controlled by the standard Timer 0 control bits. TH0 operates as an 8-bit Timer, controlled by the Timer 1 control bits.

5.9 TCON Register

The TCON register serves as the control register for Timer/Counter 0 and Timer/Counter 1. It also latches the overflow interrupt sources for T0 and T1, as well as external request interrupt sources.

The TCON register is shown in Table 5-8.

Table 5-8 TCON Register

0x8F	0x8E	0x8D	0x8C	0x8B	0x8A	0x89	0x88
TF1	TR1	TF0	TR0	IE1	IT1	IE0	IT0

Among them:

- TF1: Timer/Counter 1 overflow flag. When the highest bit of Timer/Counter 1 overflows, TF1 is set to 1 by hardware and requests an interrupt to the microcontroller core. TF1 remains set until the microcontroller core responds to the interrupt, at which point TF1 is cleared by hardware; TF1 can also be cleared by software.

- TR1: Timer 1 run control bit. This bit is set and cleared by software. When GATE is 0, setting TR1 to 1 allows Timer 1 to start counting; setting TR1 to 0 disables counting. When GATE is 1 and external interrupt 1 is high, setting TR1 to 1 allows Timer 1 to start counting; setting TR1 to 0 disables counting.
- TF0: Timer/Counter 0 overflow flag. When the highest bit of Timer/Counter 0 overflows, TF0 is set to 1 by hardware and requests an interrupt to the microcontroller core. TF0 remains set until the microcontroller core responds to the interrupt, at which point TF0 is cleared by hardware; TF0 can also be cleared by software.
- TR0: Timer 0 run control bit. This bit is set and cleared by software. When GATE is 0, setting TR0 to 1 allows Timer 0 to start counting; setting TR0 to 0 disables counting. When GATE is 1 and external interrupt 0 is high, setting TR0 to 1 allows Timer 0 to start counting; setting TR0 to 0 disables counting.
- IE1: External interrupt 1 request flag. When IE1 is set to 1, an external interrupt requests an interrupt to the microcontroller core. When the microcontroller core responds to this interrupt, IE1 is cleared by hardware.
- IT1: External interrupt 1 trigger control bit. When IT1 is 0, external interrupt 1 is low-level triggered. When external interrupt 1 inputs a low level, IE1 is set to 1. In low-level trigger mode, the external interrupt source must remain valid at a low level until the interrupt is acknowledged by the microcontroller core, and before the interrupt service routine is completed, the external interrupt source must be cleared; otherwise, another interrupt will occur. When IT1 is 1, external interrupt 1 is edge-triggered; when external interrupt 1 transitions from high to low, IE1 is set to 1, sending an interrupt request to the microcontroller core.
- IE0: External interrupt 0 request flag. When IE0 is set to 1, an external interrupt requests an interrupt to the microcontroller core. When the microcontroller core responds to this interrupt, IE0 is cleared by hardware.
- IT0: External interrupt 0 trigger control bit. When IT0 is 0, external interrupt 0 is

low-level triggered. When external interrupt 0 inputs a low level, IE0 is set to 1. In low-level trigger mode, the external interrupt source must remain valid at a low level until the interrupt is acknowledged by the microcontroller core, and before the interrupt service routine is completed, the external interrupt source must be cleared; otherwise, another interrupt will occur. When IT0 is 1, external interrupt 0 is edge-triggered; when external interrupt 0 transitions from high to low, IE0 is set to 1, sending an interrupt request to the microcontroller core.

5.10 TL0, TH0 Register

The TL0 and TH0 registers are used to store the initial value of Timer/Counter 0, saving the corresponding data based on the operating mode of the timer/counter.

5.11 TL1, TH1 Register

The TL1 and TH1 registers are used to store the initial value of Timer/Counter 1, saving the corresponding data based on the operating mode of the timer/counter.

5.12 SCON Register

The SCON register is used to select the working mode and control functions of serial communication. The SCON register is shown in Table 5-9.

Table 5-9 SCON Register

0x9F	0x9E	0x9D	0x9C	0x9B	0x9A	0x99	0x98
SM0	SM1	SM2	REN	TB8	RB8	TI	RI

Among them,

- SM0 and SM1: Serial port working mode control bits.

  The SM0 and SM1 bits control the serial port working modes, as shown in Table 5-10.

Table 5-10 SM0, SM1 Bit

SM0	SM1	Working mode	Functional description	Baud rate
0	0	Mode 0	Synchronous shift serial	SYSCLK/12

			mode.	
0	1	Mode 1	8-bit UART, variable baud rate	$(2^{SMOD}/32)\times$(Timer 1 overflow rate)
1	0	Mode 2	9-bit UART	$(2^{SMOD}/64)\times$ SYSCLK
1	1	Mode 3	9-bit UART, variable baud rate	$(2^{SMOD}/32)\times$(Timer 1 overflow rate)
When the microcontroller operates in the 12T mode, Timer 1 overflow rate = SYSCLK/12/(256-TH1). When the microcontroller operates in the 6T mode, Timer 1 overflow rate = SYSCLK/6/(256-TH1).				

- SM2: Multi-machine communication control bit. In mode 0, SM2 must be set to 0; in mode 1, RI is set to 1 only when SM2 is 1 and a valid stop bit is received; in mode 2 or mode 3, RI is set to 1 only when SM2 is 1 and the received 9th data bit RB8 is 0.
- REN: Serial port receive control bit. When REN is 1, the serial port is allowed to receive data; when REN is 0, the serial port is prohibited from receiving data.
- TB8: Transmit data bit 8. In modes 2 and 3, TB8 is the 9th data bit to be transmitted. In multi-machine communication, it indicates whether the transmission is an address or data: TB8 is 0 when transmitting data and TB8 is 1 when transmitting an address.
- RB8: Receive data bit 8. In modes 2 and 3, RB8 holds the received 9th data bit, which is used to identify the characteristics of the received data. In mode 1, if SM2 is 0, RB8 receives the stop bit; RB8 is not used in mode 0.
- TI: Serial port transmit interrupt flag. When data is sent in mode 0, TI is set to 1 by hardware after sending 8 bits of data. When sending in modes 1, 2, or 3, TI is set to 1 at the start of sending the stop bit. TI being 1 indicates that the serial port is requesting an interrupt from the microcontroller core. The microcontroller core does not clear TI when responding to the interrupt request and switching to the interrupt service routine, so TI must be cleared by the user in the interrupt service routine.

- RI: Serial port receive interrupt flag. If the serial port allows receiving and operates in mode 0, RI is set to 1 whenever the 8th data bit is received. If operating in modes 1, 2, or 3 and SM2 is 0, RI is set to 1 whenever the stop bit is received. When the serial port operates in modes 2 or 3 and SM2 is 1, RI is set to 1 only after receiving the 9th data bit RB8 as 1 and the stop bit. RI being 1 indicates that the serial port is requesting an interrupt from the microcontroller core, and RI must be cleared by the user in the interrupt service routine.

5.13 SBUF Register

The SBUF register is used to buffer the data that is to be sent or received by the serial port.

5.14 PCON Register

The PCON register is used for power control and is not bit-addressable.

The PCON register is shown in Table 5-11.

Table 5-11 PCON Register

7	6	5	4	3	2	1	0
SMOD				GF1	GF0	PD	IDL

Among them,

- SMOD: Baud rate selection bit. When SMOD is set to 1, the serial port baud rate is doubled. The default value of SMOD is 0 after a system reset.
- The OC8051 microcontroller is a soft core, so apart from the SMOD register bit, all other bits are virtual placeholders.

5.15 SP Register

The SP register is an 8-bit special-purpose register that indicates the location of the top of the stack in the on-chip data memory.

5.16 RCAP2L, RCAP2H Register

The RCAP2L and RCAP2H registers are used to store the reload/capture values for Timer/Counter 2, saving the corresponding data based on the operating mode of the timer/counter.

5.17 T2CON Register

The T2CON register is used to control the operating mode of Timer/Counter 2.

The T2CON register is shown in Table 5-12.

Table 5-12 T2CON Register

0xCF	0xCE	0xCD	0xCC	0xCB	0xCA	0xC9	0xC8
TF2	EXF2	RCLK	TCLK	EXEN2	TR2	C_T2	CP_RL2

Among them,

- TF2: Timer 2 overflow interrupt flag. TF2 must be cleared by the user program. When Timer 2 is used as a baud rate generator for the serial port, TF2 will not be set to 1.
- EXF2: External interrupt flag for Timer T2. When EXEN2 is set to 1, EXF2 is set to 1 when a negative edge occurs on T2EX, indicating an interrupt. EXF2 must be cleared by software.
- RCLK: Receive clock selection flag for the serial port. When RCLK = 1, T2 operates in baud rate generator mode.
- TCLK: Transmit clock selection flag for the serial port. When TCLK = 1, T2 operates in baud rate generator mode.
- EXEN2: External enable flag for Timer 2. When EXEN2 is set to 1 and Timer 2 is not used as the serial port clock, it allows negative edges on T2EX to generate capture or reload.
- TR2: Timer/Counter 2 control bit. When TR2 is set to 1, counting is allowed; when

TR2 is set to 0, counting is disabled.

- C/T2: External timer/counter selection bit. When C/T2 = 1, T2 acts as an external event counter; when C/T2 = 0, T2 operates as a timer, using the divided oscillator pulse signal as the counting signal.
- CP/RL2: Capture and constant auto-load mode selection bit. When set to 1, T2 operates in capture mode; when set to 0, T2 operates in constant auto-load mode. When TCLK or RCLK is set to 1, CP/RL2 is ignored.

5.18 TL2, TH2 Register

The TL2 and TH2 registers are used to store the initial values for Timer/Counter 2, saving the corresponding data based on the operating mode of the timer/counter.

6 Usage of IDE Software

6.1 Software Version

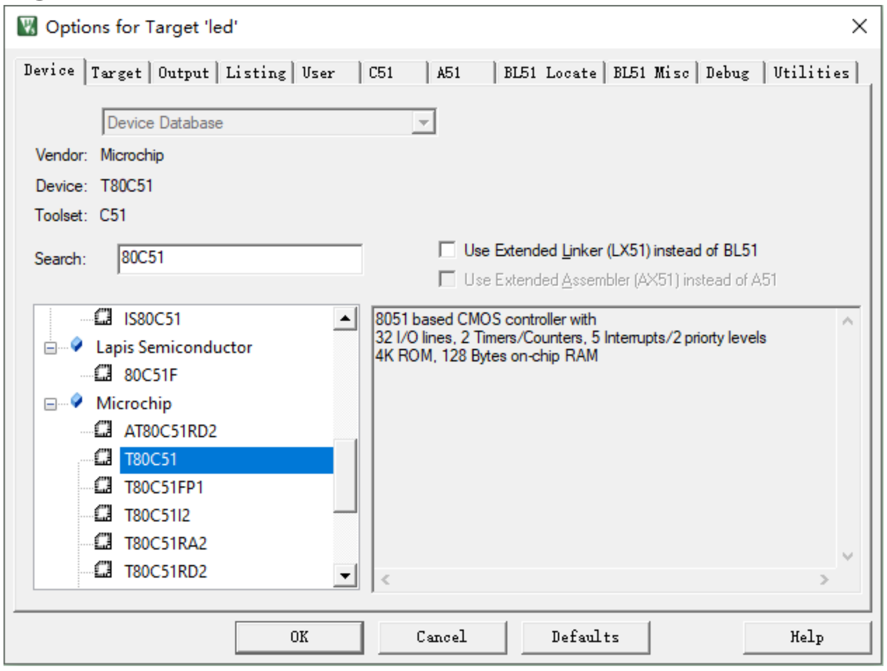
Tested software version: ARM Keil uVision5 C51 V9.24.

6.2 Configuration

6.2.1 Device Selected

When creating a new software project, you need to select the device model. Since the OC8051 is a soft core, it is sufficient to ensure that the register definitions in the hardware design are consistent with those in "reg51.h". Therefore, you can arbitrarily choose any 8051 devices during the device selection, such as the "Microchip > T80C51" selected in the reference design, as shown in Figure 6-1.

Figure 6-1 Device Selected



Options for Target 'led'

Device Database
Vendor: Microchip
Device: T80C51
Toolset: C51
Search: 80C51

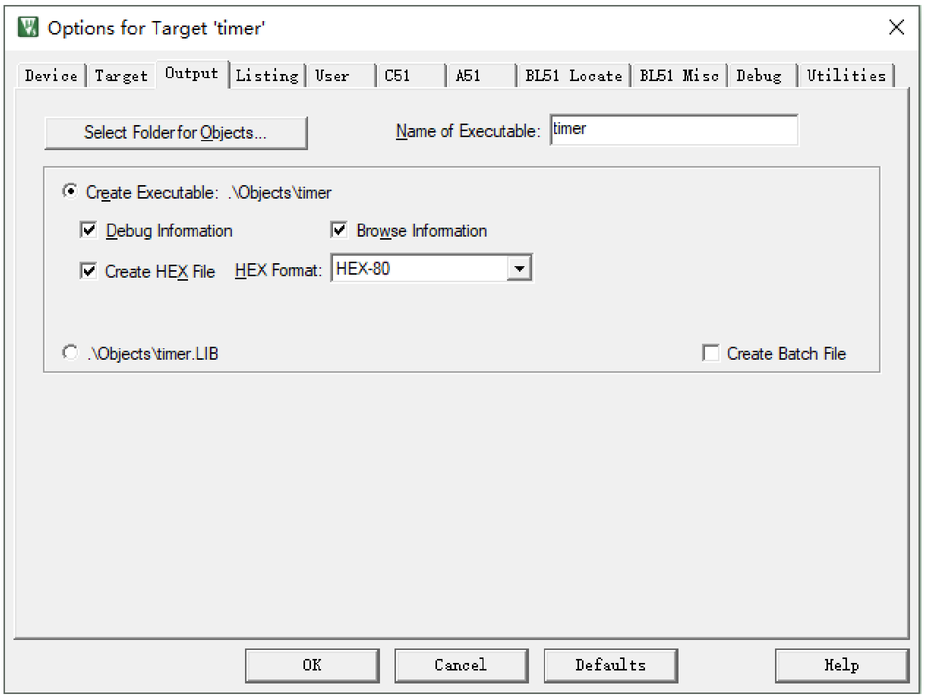
Use Extended Linker (LX51) instead of BL51
Use Extended Assembler (AX51) instead of A51

8051 based CMOS controller with
32 I/O lines, 2 Timers/Counters, 5 Interrupts/2 priorty levels
4K ROM, 128 Bytes on-chip RAM

6.2.2 File Output

Since the software tool "...\tool\oc8051_Rom_Maker.exe" requires an input ".hex" file, you need to set it to output a ".hex" file. Please check the option "Options for Target… > Output > Create HEX File" to ensure the software project outputs a ".hex" file, as shown in Figure 6-2.

Figure 6-2 File Output



Name of Executable: timer

(●) Create Executable: .\Objects\timer

[✓] Debug Information [✓] Browse Information

[✓] Create HEX File HEX Format: HEX-80

( ) .\Objects\timer.LIB [ ] Create Batch File

6.3 Compile

Compile the software project to generate the “.hex” file.

7 Reference Design

7.1 Hardware Design

7.1.1 Hardware Project

It is located in … \ ref_design \ FPGA_RefDesign \ DK_START_GW5A25_V2.0 \ oc8051.

Note:

Before compiling this project, please be sure to modify the parameter "OC8051_CODES_FILE" in the file "oc8051_defines.v" to your local actual file path.

7.1.2 Parameter Configuration

Table 7-1 Parameter Configuration

Parameter option	Description	Setting
INT_ROM_SIZE	On-chip program memory size.	4096
INT_RAM_SIZE	On-chip data memory size.	8
OC8051_CODES_FILE	Software program code file.	“led.in”
OC8051_UART	Enable UART.	√
OC8051_TC01	Enable Timer/Counter 0 and Timer/Counter 1.	√
OC8051_TC2	Enable Timer/Counter 2.	√
OC8051_PORTS	Enable I/O PORTS	√
OC8051_PORT0	Enable P0.	√
OC8051_PORT1	Enable P1.	√
OC8051_PORT2	Enable P2.	√
OC8051_PORT3	Enable P3.	√
OC8051_ROM	Enable on-chip program memory.	√
OC8051_GENERIC	Enable on-chip program memory.	√

7.1.3 Test Environment

- Gowin_V1.9.10.02 (64-bit)
- DK_START_GW5A25 V2.0

  GW5A-LV25UG324C2/I1

  GW5A-25 A

7.2 Software Design

7.2.1 Running LED

7.2.1.1 Software Project

It is located in …\ref_design\MCU_RefDesign\led.

7.2.1.2 Functional Description

Constrain the output ports P0^0, P0^1, P0^2, and P0^3 of the OC8051 microcontroller to four LEDs. The LED running light effect can be achieved by assigning values to the P0 register to make the LEDs blink in sequence.

In this example, the LEDs connected to P0.0 to P0.3 will blink on and off with a delay. You can modify the P0 register values to create different patterns or effects.

7.2.1.3 Software Code

The software code is as follows:

```
#include <reg51.h>

void main()
{
        unsigned int i;
        unsigned int k;
        while(1)
        {
                for(i=0;i<4;i++)
                {
                        k=50000;
                        P0=(0x01<<i);
                        while(k--);
                }
        }
}
```

Note:

- If a delay is implemented using a `for` loop, the number of iterations in a single `for` loop should not exceed 32,767.

7.2.2 Button

7.2.2.1 Software Project

It is located in …\ref_design\MCU_RefDesign\key.

7.2.2.2 Functional Description

Constraint the P1.0 input port of the OC8051 microcontroller to a button, and the P0.0 output port to an LED. The LED's level will toggle when the button is pressed.

7.2.2.3 Software Code

The software code is as follows:

```
#include <reg51.h>

sbit KEY=P1^0;
sbit LED=P0^0;

void delay(unsigned int t);

void main()
{
	while(1)
	{
		if(KEY == 0)
		{
			delay(50000);		//Debouncing with a delay
			if(KEY == 0)
			{
				LED = ~LED;
				while(KEY == 0); //To wait for the button to be released
			}
		}
	}
}

void delay(unsigned int t)
{
	while(t--);
}
```

Note:

- During the button press, bouncing may occur, leading to multiple signal transitions. Therefore, a delay function is needed for debouncing.

7.2.3 External Interrupt

7.2.3.1 Software Project

It is located in …\ref_design\MCU_RefDesign\ext_inter.

7.2.3.2 Functional Description

Constraint the external interrupt 1 input port of the OC8051 microcontroller to a button, and the P0.0 output port to an LED. The LED's level will toggle by triggering the interrupt with the button press.

7.2.3.3 Software Code

The software code is as follows:

```
#include <reg51.h>

sbit LED=P0^0;

void ext_init();

void ext_init()
{
        EA = 1;
        IT0 = 1;
        EX0 = 1;
}

void main()
{
        LED=0;
        ext_init();
        while(1)
        {
        }
}

void ext_inter() interrupt 0
{
        LED = !LED;
}
```

Note:

- Before enabling the interrupt for external interrupt 1, the global interrupt enable must be activated.
- The interrupt service routines for external interrupt 1 and external interrupt 2 are `interrupt 0` and `interrupt 2`, respectively.

7.2.4 Timer 0

7.2.4.1 Software Project

It is located in …\ref_design\MCU_RefDesign\timer.

7.2.4.2 Functional Description

Constraint the P0.0 output port of the OC8051 microcontroller to an LED, and implement LED level toggling using a timer overflow.

7.2.4.3 Software Code

The software code is as follows:

```
#include <reg51.h>

sbit LED=P0^0;

void Timer_Init();

void Timer_Init()
{
        TMOD = TMOD & 0xF0;
        TMOD = TMOD | 0x01;//Enable Timer 0 and working mode 1
        TF0 = 0;                        //Clear overflow flag
        TH0 = (65535-50000)/256;
        TL0 = (65535-50000)%256;
        TR0 = 1;                        //Start timer
}

void main()
{
        unsigned int i;

        LED=0;

        Timer_Init();

        while(1)
        {
            i=0;
            LED = !LED;
            while(i<10)
            {
                while(TF0==0);
                TH0=(65536-50000)/256;
                TL0=(65536-50000)%256;
                TF0=0;
                i++;
            }
        }
}
```

Note:

- After reloading the initial value, TR0 must be set to 1 to start the timer.
- After a timer overflow, the overflow flag TF0 needs to be cleared in software. If you want to continue using the timer after the overflow, you must reload the initial value again.

7.2.5 Counter 0

7.2.5.1 Software Project

It is located in …\ref_design\MCU_RefDesign\counter.

7.2.5.2 Functional Description

Constraint the external input pin of Timer/Counter 0 on the OC8051 microcontroller to one header pin, and constrain the P0.4 output port to another header pin. Connect these two header pins with a jumper wire. Additionally, constrain the P0.0 output port to an LED, and implement LED level toggling based on the parity of the counter.

7.2.5.3 Software Code

The software code is as follows:

```
#include <reg51.h>

sbit LED=P0^0;
sbit WIRE=P0^4;

void Timer_Init();

void Timer_Init()
{
	TMOD = TMOD & 0x00;
	TMOD = TMOD | 0x15;	//Enable Timer 1, working mode 1. Counter 0, working mode 1.
	TF1 = 0;
	TH1=(65536-50000)/256;
	TL1=(65536-50000)%256;
	TR1 = 1;
	TF0 = 0;
	TH0 = 0;
	TL0 = 0;
	TR0 = 1;
}

void main()
{
	unsigned int i;

	LED=0;
	WIRE=0;

	Timer_Init();

	while(1)
	{
		i=0;
		WIRE = !WIRE;
		while(i<10)
		{
			while(TF1==0);
			TH1=(65536-50000)/256;
			TL1=(65536-50000)%256;
```

```
            TF1=0;
            i++;
        }
        if(TL0%2==1)
        {
            LED = !LED;
        }
    }
}
```

Note:

- Use Timer/Counter 1 in timer mode to generate a level toggle on the P0.4 pin, while using Timer/Counter 0 in counter mode to receive the pulses generated from the P0.4 pin.

7.2.6 Timer 0 Interrupt

7.2.6.1 Software Project

It is located in …\ref_design\MCU_RefDesign\timer_inter.

7.2.6.2 Functional Description

Constraint the P0.0 output port of the OC8051 microcontroller to an LED, and implement LED level toggling by triggering a timer interrupt.

7.2.6.3 Software Code

The software code is as follows:

```
#include <reg51.h>

sbit LED=P0^0;

void Timer_Init();

void Timer_Init()
{
    TMOD = TMOD & 0xF0;
    TMOD = TMOD | 0x01;
    TF0 = 0;
    EA = 1;                 //Enable global interrupt
    ET0 = 1;                //Enable Timer 0 interrupt
    TH0 = (65535-50000)/256;
    TL0 = (65535-50000)%256;
    TR0 = 1;                //Start Timer 0
    PT0 = 0;                //Timer 0 interrupt priority is lower
}

void main()
{
    Timer_Init();

    while(1)
    {
    }
}

void Timer0_Inter() interrupt 1
{
    static unsigned int T0Count;
    TF0 = 0;
    TH0 = (65535-50000)/256;
    TL0 = (65535-50000)%256;
    T0Count++;
    if(T0Count >= 20)
    {
        T0Count = 0;
        LED = !LED;
    }
}
```

Note:

- Before enabling the interrupt for Timer 0, the global interrupt enable must be activated.
- The interrupt service routines for Timer 0 and Timer 1 are defined as `interrupt 1` and `interrupt 3`, respectively.

7.2.7 Timer 2

7.2.7.1 Software Project

It is located in …\ref_design\MCU_RefDesign\timer2.

7.2.7.2 Functional Description

Constraint the P0.0 output port of the OC8051 microcontroller to an LED, and implement LED level toggling using a timer overflow.

7.2.7.3 Software Code

The software code is as follows:

```
#include <reg51.h>

sbit LED=P0^0;

sfr RCAP2H=0xCB;
sfr RCAP2L=0xCA;

sfr T2CON=0xC8;
sbit TF2=T2CON^7;
sbit TR2=T2CON^2;

sfr TH2=0xCD;
sfr TL2=0xCC;

sbit ET2=IE^5;

void Timer2_Init();

void Timer2_Init()
{
	TH2=(65536-50000)/256;
	TL2=(65536-50000)%256;
	TF2=0;
	TR2=1;
}

void main()
{
	unsigned int i;

	LED=0;

	Timer2_Init();

	while(1)
```

```
    {
        i=0;
        LED = !LED;
        while(i<10)
        {
            while(TF2==0);
            TH2=(65536-50000)/256;
            TL2=(65536-50000)%256;
            TF2=0;
            i++;
        }
    }
}
```

Note:

- Since the `reg51.h` header file does not include registers related to Timer/Counter 2, you need to define the relevant registers in your software project.

7.2.8 Timer 2 Interrupt

7.2.8.1 Software Project

It is located in …\ref_design\MCU_RefDesign\timer2_inter.

7.2.8.2 Functional Description

Constraint the P0.0 output port of the OC8051 microcontroller to an LED, and implement LED level toggling by triggering a timer interrupt.

7.2.8.3 Software Code

The software code is as follows:

```
#include <reg51.h>

sbit LED=P0^0;

sfr RCAP2H=0xCB;
sfr RCAP2L=0xCA;

sfr T2CON=0xC8; //Timer/Counter 2 control register
sbit TF2=T2CON^7; / Timer/Counter 2 overflow flag bit
sbit TR2=T2CON^2; // Timer/Counter 2 running flag bit
```

```
sfr TH2=0xCD; // Timer/Counter 2 data register high 8-bit
sfr TL2=0xCC; // Timer/Counter 2 data register low 8-bit

sbit ET2=IE^5; // Timer/Counter 2 interrupt flag bit

void Timer2_Init();
void timer2_inter();

void Timer2_Init()
{
	TH2=(65536-50000)/256;
	TL2=(65536-50000)%256;
	TF2=0;
	TR2=1;
	ET2=1;
	EA=1;
}

void main()
{
	LED=0;

	Timer2_Init();

   while(1)
	{
	}
}

void timer2_inter() interrupt 5 using 1
{
	static unsigned int T0Count;
	TF2=0;
	TH2 = (65535-50000)/256;
	TL2 = (65535-50000)%256;
	T0Count++;
	if(T0Count >= 10)
	{
		T0Count = 0;
		LED = !LED;
	}
}
```

Note:

- Before enabling the interrupt for Timer 2, the global interrupt enable must be activated.
- The interrupt service routine for Timer 2 is defined as `interrupt 5`.
- You can use `using n` to access different register banks.

7.2.9 Serial Communication

7.2.9.1 Software Project

It is located in …\ref_design\MCU_RefDesign\uart.

7.2.9.2 Functional Description

Constraint the TXD and RXD ports of the OC8051 microcontroller to header pins, and use a PC serial debugging tool to receive strings sent from the OC8051 microcontroller's serial port.

7.2.9.3 Software Code

The software code is as follows:

```
#include <reg51.h>

void SendString (unsigned char *str);
void SendBit (unsigned char UART_data);

void UART_Init (void)
{
	EA = 1; // Enable global interrupts (if interrupts are not used, you can use `//` to comment it out).
	ES = 0; //Enable serial interrupt.
	TMOD = 0x20; //Timer T/C1 working mode 2.
	SCON = 0x50; //Serial working mode 1，enable serial receives (when SCON = 0x40, disable serial receiving).
	TH1 = 0xF3; //The high 8-bit setting of timer initialized value.
	TL1 = 0xF3; //The low 8-bit setting of timer initialized value.
	PCON = 0x80; //To implement baud rate doubling (comment out this line for a baud rate of 2400).
	TR1 = 1; //Start timer.
}

void SendString (unsigned char *str)
{
        while(*str != '\0')
        {
                SendBit(*str);
                str++;
        }
        *str = 0;
}

void SendBit (unsigned char UART_data)
{
        SBUF = UART_data; //Place the data to be sent into the serial port buffer register (SBUF).
        while(TI == 0); //Check the transmit interrupt flag.
        TI = 0; //Clear the transmit interrupt flag by setting it to 0 in software.
}


void main()
{
        unsigned char *s = "hello world\r\n";
        unsigned int i=0;

        UART_Init();

        while(1)
        {
                i = 50000;
                SendString(s);
                while(i--);
        }
}
```

Note:

- The baud rate calculation method can be referenced in Section 5.12. To ensure reliable transmission, the baud rate error is required to be no greater than 2.5%.
- If the serial interrupt is enabled and string transmission is performed, the first byte of the string will be transmitted twice during the first transmission.

7.2.10 Serial Interrupt

7.2.10.1 Software Project

It is located in …\ref_design\MCU_RefDesign\uart_inter.

7.2.10.2 Functional Description

Constraint the TXD and RXD ports of the OC8051 microcontroller to header pins. Use a PC serial debugging tool to send strings to the OC8051 microcontroller's serial port and receive the strings that are echoed back from the OC8051 microcontroller's serial port.

7.2.10.3 Software Code

The software code is as follows:

```
#include <reg51.h>

void UART_Init (void);



void UART_Init (void)
{
        EA = 1; // Enable global interrupts (you can comment this out with `//` if interrupts are not
used).
        ES = 1; //Enable the UART serial interrupt.
        TMOD = 0x20; //To set Timer/Counter (T/C1) to operate in mode 2.
        SCON = 0x50; //To configure the serial port to work in mode 1 and enable serial reception,
you can set the SCON register accordingly.
        TH1 = 0xF3; //The high 8-bit setting of timer initialized value.
        TL1 = 0xF3; //The low 8-bit setting of timer initialized value.
        PCON = 0x80; // o implement baud rate doubling (comment out this line for a baud rate of
2400).
        TR1 = 1; //Start timer
}

void main()
{
        UART_Init();

        while(1)
        {
        }
}

void UART_Inter (void) interrupt 4
{
        unsigned char UART_data; //Define a variable for receiving data from the serial port.
        if(RI != 0)
        {
                RI = 0; // Clear the receive interrupt flag by setting it to 0 in software.
                UART_data = SBUF; //Store the received serial data into a variable named
`UART_data`.
                SBUF = UART_data;
        }
        else if(TI != 0)
        {
                TI = 0;
        }
}
```

Note:

- Both the receiving and sending interrupts of the serial port are handled in the same interrupt routine.
- The interrupt service routine for the serial port is `interrupt 4`.

8 Development Process

The development process for the Gowin OC8051 microcontroller is as follows:

1. Obtain and install the Gowin® YunYuan Software, Gowin_Vx.x.
2. Obtain and install the ARM Keil C51 Vx.x software.
3. Obtain the software development toolkit for the Gowin OC8051.
4. Create a new or open an existing software project in ARM Keil C51 (for example, …\ref_design\MCU_RefDesign\led), configure and compile it to generate the ".hex" file.
5. Use the oc8051_Rom_Maker.exe tool (…\tool\oc8051_Rom_Maker.exe) to input the .hex file and generate the .in file.
6. In the YunYuan software, create a new or open an existing hardware project (for example, …\ref_design\FPGA_RefDesign\DK_START_GW5A25_V2.0\oc8051), set the "OC8051_CODES_FILE" parameter in the "oc8051_defines.v" file to the ".in" file generated in step 5 as the initial value of the on-chip program memory; then configure, synthesize, place, and route to generate the bitstream file.
7. If there is serial output, please open the upper computer serial debugging assistant software, and set the baud rate to 4800.
8. Use the Programmer tool to download the bitstream file.