Cmsemicon 8051 Microcontrollers

Application Notes

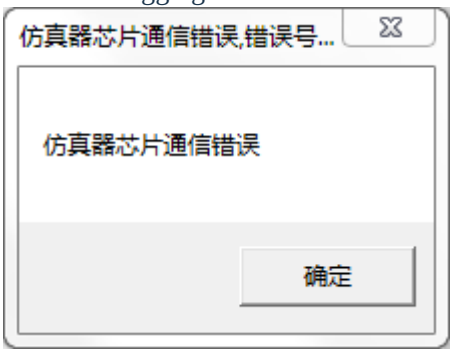
Rev. 1.1.0

Table of Contents

The application notes apply to the following Cmsemicon 8051 microcontroller series: CMS8S3660, CMS8S3680, CMS8S6980, CMS8S6990, CMS8S5880, CMS8S5887, CMS8S5888, CMS8S5889, CMS80F231x, CMS80F253x, CMS80F261x, CMS80F251x, CMS80F751x, CMS8S7885, CMS8S7895, CMS8S6997, CMS8S6998, CMS8S6999, CMS8S589x, CMS8S369x, CMS8M35xx, and other series.

1. Simulation Usage Notes

- When configuring the simulation mode, it is recommended not to connect an external pull-down resistor to the DSDA pin.
- If the LED/LCD SEG corresponding to the debug port is enabled, the debug mode will be disabled.
  - For CMS80F253x and CMS80F751x series chips, the LED/LCD SEG function takes precedence over the debug function. The chip will prioritize processing the LED/LCD SEG. When debugging, please disable the SEG enable on the debug port (default is disabled). (For CMS80F261x series chips, the debug function takes precedence over the LED/LCD SEG function.)
- If a WDG reset or register reset occurs before the DEBUG RST, the chip will not STOP after the DEBUG RST.
  - If you encounter this issue, please download the latest tools to avoid this problem.
- If the program enables the LSE_TIMER timing function, the timer will not be controlled by simulation-related commands (such as step, full speed, etc.) during simulation.
  - This is a normal phenomenon as the clock keeps running continuously. Similar modules include UART and PWM.
- If the following dialog box appears, check if the debug function has been disabled. If an error occurs when downloading the program, switch the HSI frequency and then try downloading the program again. This should typically resolve the issue for debugging.

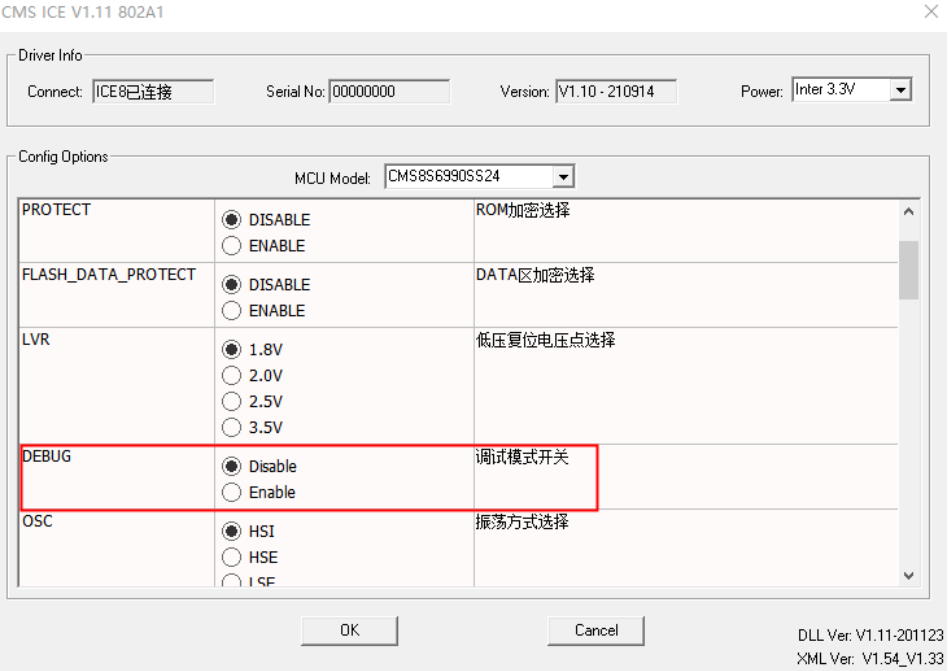


2. BANK

When using the CMS80F261x, CMS80F251x, or CMS80F751x series chips, if different BANKs are used and there is a BANK switch operation in the interrupt service routine, please save the BANK value before entering the interrupt and restore it upon exiting the interrupt (This is to prevent inconsistencies in the BANK state after exiting the interrupt.)

3. Low Power Consumption

- When using low power mode, please disable the debug function. Unused I/O pins and I/O pins that are not packaged should be configured as output with a high-level state; otherwise, the power consumption will not decrease. If the system is awakened by an interrupt, the global interrupt enable must be activated, along with the interrupt enable for the corresponding wake-up interrupt source.

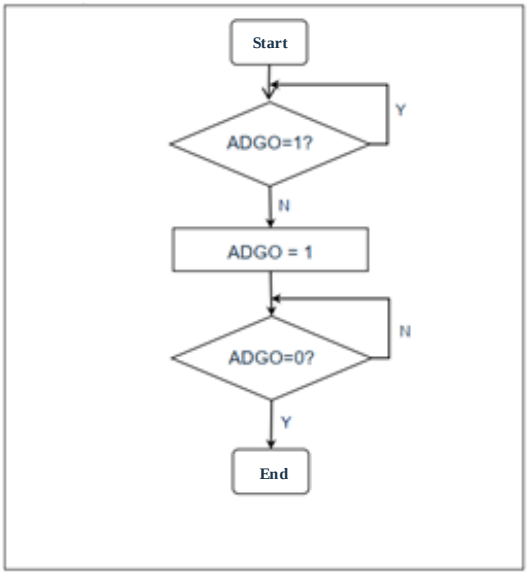


- When using the CMS8S6990, it is recommended to add a 10ms delay after the system wakes up from sleep.
- When using LSE to wake up from sleep mode, please clear the LSE count enable bit before entering sleep, and then re-enable it (clear the LSE count value).

4. ADC

When using the ADC on the 51 series chips, the following points need to be noted:

1) If you encounter insufficient sampling accuracy, it may help to reduce the ADC speed or decrease the impedance of the ADC sampling circuit.
2) When using hardware trigger for the ADC, the time interval between two adjacent trigger signals should be greater than the ADC conversion time, to avoid triggering signals during the ADC conversion process.
3) When both software and hardware triggers are used for the ADC in the program, the hardware trigger should be disabled before the software trigger initiates the ADC conversion.
4) During the ADC conversion process, all hardware and software trigger signals within this time period will be ignored.
5) When using the ADC software trigger, the following flowchart should be followed.



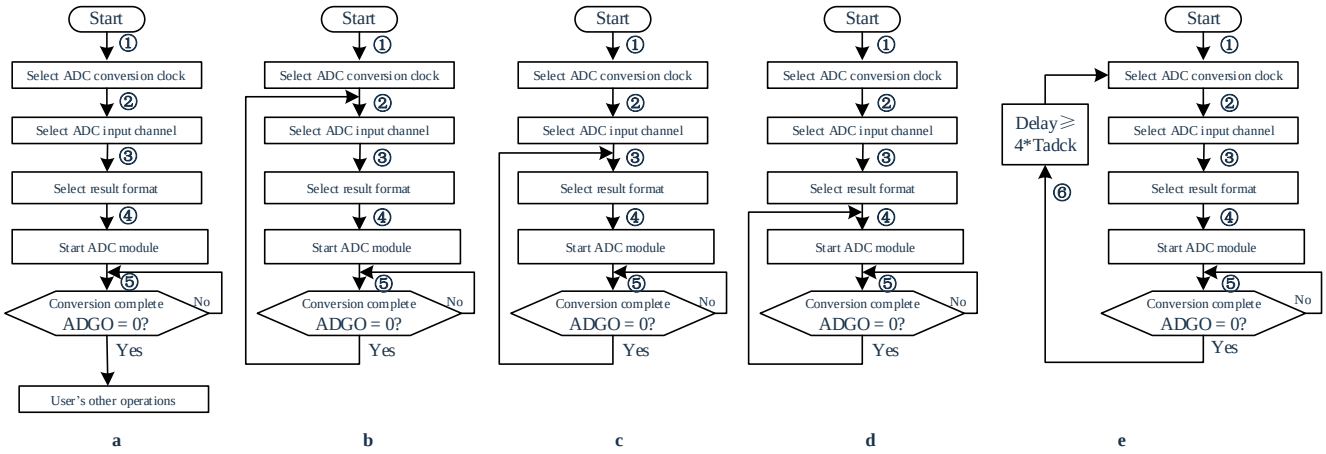
The reference code is as follows:

```
if(!ADC_IS_BUSY)                    //Check if ADC is converting
{
        _nop_();
        ADC_GO();                   //Software trigger ADC conversion
        while(ADC_IS_BUSY);         //Wait for ADC conversion to complete
}
```

```
if(!ADC_IS_BUSY)                    //Check if ADC is converting
{
        _nop_();
        ADC_DisableHardwareTrig();  //Disable hardware trigger
        ADC_GO();                   //Software trigger ADC conversion
        while(ADC_IS_BUSY);         //Wait for ADC conversion to complete
        ADC_EnableHardwareTrig();   /Enable hardware trigger
}
```

6) ADC Clock Division Ratio Operation Notes: After detecting the completion of the previous ADC conversion, if it is necessary to change the ADC clock division ratio again, it is recommended to wait for at least 4 ADC conversion clock cycles (Tadck). The details are as follows:

- ✓ As shown in Step ⑥: After detecting the completion of the previous ADC conversion, if the ADC clock division ratio needs to be changed again, it is recommended to wait for at least 4 ADC conversion clock cycles (Tadck).
- ✓ As shown in a, b, c, d: When using the ADC again, as long as the ADC clock division ratio is not modified, no delay is required.



5. I²C

- When using the CMS80F261x chip and configuring the IO port alternate function, setting PS_SCL and PS_SDA as write-only registers, the registers must be directly assigned values. The use of logical operations (AND/OR) in the program is prohibited.
- When the IO port is used as SCL/SDA, the internal pull-up resistor and open-drain settings should be referenced in the table below:

Function	Function description	Pull-up Resistor Control Register (PxUP)	Open-Drain Control Register (PxOD)	Pull-up Resistor Control Register (PxUP)	Open-Drain Control Register (PxOD)
SCL	I2C clock input/output	Controlled by PxUP to enable or disable the pull-up	Forced to enable open-drain, independent of PxOD configuration	Controlled by PxUP to enable or disable the pull-up	Forced to enable open-drain, independent of PxOD configuration
SDA	I2C data input/output	Controlled by PxUP to enable or disable the pull-up	Forced to enable open-drain, independent of PxOD configuration	Controlled by PxUP to enable or disable the pull-up	Forced to enable open-drain, independent of PxOD configuration
Specific chip models (operations may vary for different chips):		CMS8S3660, CMS8S5880, CMS8M35xx, CMS8S3680, CMS8S6980, CMS8S6990, CMS80F231x, CMS80F261x, CMS80F251x, CMS80F751x, CMS8S78xx, CMS8S588x, CMS8S5887, CMS80F253x		CMS80F761x, CMS8S6998, CMS80F262x, CMS8S369x, CMS8S589x, CMS80F161x, CMS8S6997_8_9	

6. SPI

SPI only supports 8-bit, MSB (Most Significant Bit first) mode.

When using the CMS80F261x chip and configuring the IO port alternate function, setting PS_SCLK, PS_MOSI, PS_MISO, and PS_NSS as write-only registers, the registers must be directly assigned values. The use of logical operations (AND/OR) in the program is prohibited.

7. UART

- The PS_RXDI register of the CMS80F261x chip is a write-only register. When assigning a value to this register, direct assignment must be used. The use of logical operations (AND/OR) in the program is prohibited.
- When the IO port is used as RX/TX, the internal pull-up resistor and open-drain settings should be referenced in the table below:

Function	Function description	Pull-up Resistor Control Register (PxUP)	Open-Drain Control Register (PxOD)	Pull-up Resistor Control Register (PxUP)	Open-Drain Control Register (PxOD)
TXD	UART data output	Not controlled by PxUP	Not controlled by PxOD	Not controlled by PxUP	Not controlled by PxOD
RXD	UART data input, asynchronous mode	Not controlled by PxUP	Not controlled by PxOD	Controlled by PxUP to enable or disable pull-up	Not controlled by PxOD
	UART synchronous mode data input/output	Forced pull-up, not controlled by PxUP	Not controlled by PxOD	Forced pull-up, not controlled by PxUP	Not controlled by PxOD
Specific chip models		CMS8S3660, CMS8S5880, CMS8M35xx, CMS8S3680, CMS8S6980, CMS8S6990, CMS80F231x, CMS80F261x, CMS80F251x, CMS80F751x, CMS8S78xx, CMS8S588x, CMS8S5887, CMS80F253x, CMS80F731x		CMS80F761x, CMS8S6998, CMS80F262x, CMS8S369x, CMS8S589x, CMS80F161x, CMS8S6997_8_9	

8. IO Ports

- When certain functions of the IO ports require multiple IO ports to be set as a group to work together, please pay attention to the specific usage and the interrelationship between each IO port within the group.
- When using the chip, please carefully read the chip's datasheet. If the usage conditions of the chip exceed the maximum or minimum values specified in the datasheet, it is considered to be operating outside of the specified range, and the operating performance is not guaranteed.

9. WDT

When modifying the WDCON register, there is a logical sequence that needs to be followed: first, write 0xAA and 0x55 to the TA register, and then assign a value to WDCON. When using C language for definition, it is recommended to define this operation sequence as a macro, and then directly call the defined macro wherever needed. Before writing the TA sequence, disable interrupts by setting EA=0. After the operation is completed, enable interrupts again by setting EA=1, according to the actual situation.

```
TA = 0xAA;
TA = 0x55;
WDCON = 0x80;
```

For example: #define __system_software_reset() TA=0xAA; TA=0x55;WDCON=0X80; Then, call __system_software_reset(); wherever needed.

```
//        TA = 0xAA;
//        TA = 0x55;
//        WDCON = 0x80;

      __system_software_reset();
```

10. Touch

The startup time of the touch module is 25μs. Please ensure that the startup is completed before processing further.

11. Interrupts

This section primarily discusses the precautions for using interrupts. The chip's interrupt clearing methods are: automatic hardware clearing after entering the interrupt service routine, software clearing, and read/write register clearing.

11.1 Cleared by Hardware

The interrupt flags generated by INT0 (external interrupt 0), INT1 (external interrupt 1), T0 (timer 0), T1 (timer 1), T3 (timer 3), and T4 (timer 4) will be automatically cleared by the hardware once the system enters the interrupt service routine.

Conditions for hardware automatic clearing of interrupt flags:

1) The global interrupt (EA=1) must be enabled.
2) The corresponding module's interrupt must be enabled.
3) An interrupt must be generated and the system must enter the interrupt service routine.

Therefore, the interrupt flag obtained within the interrupt service routine will always be 0.

If the interrupt enable is disabled, the flags of the above modules can also be cleared by software writing 0.

11.2 Cleared by Software

There are flags in the system that can only be cleared by software. These flags will not be automatically cleared after entering the interrupt service routine and need to be cleared by writing 0 via software. Otherwise, after exiting the interrupt service routine, the system may re-enter the interrupt service routine.

When performing software clearing, note that when multiple interrupt flags are located in the same register (such as CNIF in the comparator module, PWMPIF, PWMZIF, PWMUIF, etc. in the PWM module), and the timing of these flags is related, it is not recommended to use the read-modify-write operation. It is recommended to directly write 0, and for other unrelated flags, write 1. For example, to clear bit0: PWMUIF = 0xFE;

This operation will have no practical effect on writing 1 to unrelated interrupt flags.

11.3 Cleared by Register Read/Write

There are flags in the system that cannot be cleared by writing 0 to them, but rather require reading/writing other registers to clear the flags. For example, in the SPI interrupt flag register, the transfer complete flag (SPISIF) is set to 1. To clear it, the system must first read SPSR, then read/write SPDR to clear it.

12. External Oscillator

When using an external crystal oscillator with the chip, the following two points need to be noted:

1. The crystal oscillator should not exceed 16 MHz.
2. When using FLASH with an external oscillator as the main frequency, some chips require writing 256 bytes of 0xFF to the FLASH after each erase operation. For detailed operations, please refer to Chapter 14: Flash Usage. (For specific chips that require this operation, refer to the detailed description in Chapter 14.)

Sector Erase Operation	Continuous writing of 256 bytes to Flash	Read, write, and erase operations on the same sector

If a sector is being erased, it is necessary to wait for the erasure to complete before performing operations on other sectors.

13. Data Flash

- The description of DATA FLASH is provided in the chip reference manual or user manual. The usage and operations of Data Flash are the same as those of general FLASH, except that the 4th bit (MREG) of the FLASH memory control register MCTRL is used to select whether to operate on Data Flash (set to 1) or Program Flash (set to 0).

  Bit 4 MREG: Flash area selection bit
  1= Select data area (lower 10 bits of the address are valid)
  0= Select program area (lower 16 bits of the address are valid)

- When calling the FLASH library, the library functions use conditional statements to set the MREG register of MCTRL. However, if the program only needs to operate on DATA FLASH, it is recommended to set MREG using the method shown on the right side of the diagram below.

```
if(codeordata==0)
  MCTRL = 0X19;
else
  MCTRL = 0X09;
```

Not Recommended

```
MCTRL = 0X09;
```

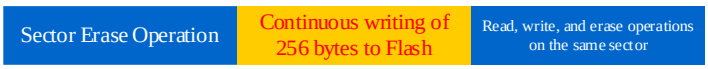
Recommended

- After each FLASH operation, it is necessary to include nop6(); to wait for the completion of the FLASH operation (include 6 NOP() instructions).
- For CMS80F261x series chips, when the program write protection is set for the 0-4K memory space, write operations to the DATA FLASH area are not allowed.
- For CMS8M35xx, CMS8S3680, CMS8S6980, CMS8S6990, and CMS80F231x series chips, when the program write protection is set for the 0-2K memory space, write operations to the DATA FLASH area are not allowed.

14. FLASH Usage

14.1 FLASH Erase Operation

For certain chips, after each FLASH erase operation, it is necessary to write 256 bytes of 0xFF consecutively to the FLASH memory (the write address can be any unused address in the FLASH, or it can be any address within the erased sector). The specific operation is as follows:



Operation details:

```
addr=0x1000;
FLASH_Erase(FLASH_CODE,addr);                  // The sector erase time is approximately: 4.6ms
for(i=0;i<256;i++)                             // The time interval for writing 256 bytes consecutively to Flash
{
  FLASH_Write(FLASH_CODE,addr, 0xFF);          // The write address can be any address within this sector.
}
```

The chips that require this operation are as follows:

CMS8M35xx, CMS8S3680, CMS8S6990, CMS8S5885, CMS80F231x, CMS80F261x, CMS80F251x, CMS80F751x, CMS8S78xx, CMS8S588x, CMS8S5887 and CMS80F253x.

14.2 FLASH Locking

Before performing operations on the FLASH, it is necessary to unlock the FLASH, and after completing the operation, the FLASH should be locked.

- For chips such as CMS8S3660, CMS8S3680, CMS8S5880, CMS8S6980, CMS8S6990, CMS80F261x, CMS8M35xx, etc., the MLOCK register can be operated directly when unlocking the FLASH.

0xFB	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
MLOCK	MLOCK7	MOCK6	MLOCK5	MLOCK4	MLOCK3	MLOCK2	MLOCK1	MLOCK0
R/W	W	W	W	W	W	W	W	W
Reset value	0	1	0	1	0	1	0	1

Bit7~Bit0	MLOCK<7:0>:	Memory operation enable bit (this register supports write operation only)
	AAH=	R/W/E operations related to Memory are enabled
	00H/FFH/55H=	Operations are disabled
	Other =	Settings are disabled

For example: MLOCK = 0xaa, unlock the FLASH; MLOCK = 0x55, lock the FLASH.

- For chips such as CMS8S5887, CMS8S5888, CMS8S5889, CMS80F231x, CMS80F253x, CMS80F251x, CMS80F751x, CMS8S7885, CMS8S7895, CMS8S6997, CMS8S6998, CMS8S6999, CMS8S589x, CMS8S369x, etc., to unlock the FLASH, you need to write the TA sequence. The TA sequence is as shown below:

```
MOV    TA,#0AAH
MOV    TA,#055H
MOV    MLOCK,#0AAH
```

Note: During the write operation, no other operations, including interrupt operations, should be inserted, otherwise the MLOCK operation will fail. There are two suggested methods to handle this:

a. Define the following statements as macro instructions to be called:

#define MLOCK_Enable() TA=0xAA, TA=0x55, MLOCK=0xAA

b. Disable interrupts (EA = 0) before writing the TA sequence, and re-enable interrupts (EA = 1) after the operation is complete.

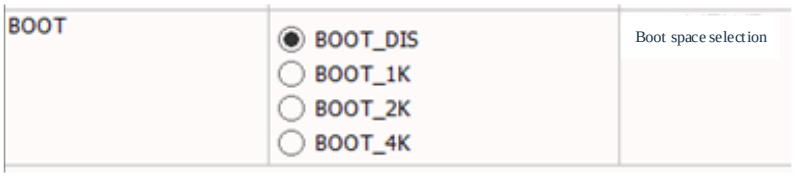
14.3 FLASH CRC

When performing a CRC on the FLASH, it is important to note that after the process is complete, the CRCASEL bit in the MSTATUS register must be cleared. Otherwise, the chip will continue to perform read and write operations on the FLASH.

15. Boot Usage

During program execution, if the pointer is directed to a non-existent address or an address that exceeds the chip's address range (for example, when using the BOOT function, and the next execution address points to an address outside the BOOT region), this is not allowed.

The size of the FLASH program space varies between different chips. The FLASH program space is divided into the BOOT region and the APROM region. The size of the BOOT region is allocated by the user configuration register, and this can be selected in the KEIL Project options, as shown in the figure below:



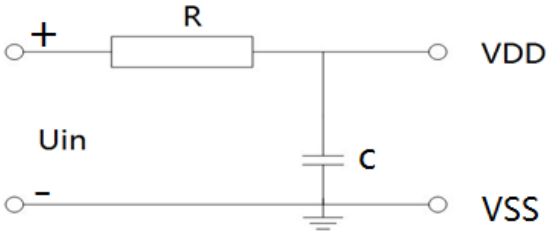
Since the BOOT region is part of the FLASH, its usage must also follow the precautions outlined in Chapter 14 for using FLASH (unlocking, write timing, erase intervals, erase time, etc.).

16. Chip Operating Voltage

When designing the power supply circuit for the chip, it is important to pay attention to the chip's reset voltage curve, power-up curve, and other related conditions.

For the chip power supply circuit design (especially for the CMS8S5880):

- When the chip operates within a voltage range of 2.1V to 4.5V, a 0.1μF (104) capacitor can be placed in parallel between the VDD and VSS pins.
- If the chip needs to operate within a voltage range of 4.5V to 5.5V, a 4.7Ω to 5.1Ω resistor can be placed in series with the power supply circuit, and a 2.2μF to 4.7μF capacitor can be placed in parallel with VSS to form an RC filter circuit (as shown in the diagram below).



RC Filter Circuit

17. Cautions for TA Timing Operations

When operating certain registers, it is necessary to first write the TA timing. Only after the TA timing is completed will the operations on the registers be effective. The TA write timing requires that the entire writing process must not be interrupted; otherwise, the TA write timing will not succeed.

Before writing the TA timing, ensure that interrupts are disabled. Once the write sequence is completed, interrupts can be enabled again as needed.

For the 8051 chip, operations on the WDCON, CLKDIV, SCKSEL, MLOCK, WWCON0, WWCON1, and WWCMPD registers require writing the TA timing (the exact registers are based on the actual product).

You can also define the write timing as a macro, for example:

#define MLOCK_Enable() TA=0xAA, TA=0x55, MLOCK=0xAA;

18. Operational Amplifier

18.1 Overview

The operational amplifier (op-amp) is a general-purpose op-amp built into the MCU. The input and output terminals of the amplifier are open, and the gain of the op-amp can be adjusted by external resistors to achieve basic signal amplification and operation functions.

18.2 Application Notes for the Built-in Operational Amplifier

1. The output current of the built-in op-amp is 2mA, so the input resistance and feedback resistance should be as large as possible to minimize the op-amp's own losses. (For example, in a non-reverse amplifier circuit with a 10x gain, an input resistance of 2K and a feedback resistance of 20K can be selected.)
2. The output voltage of the built-in op-amp must be at least 350mV to ensure normal operation. Therefore, for amplifying small signals, a DC bias should be added to the input.
3. The built-in op-amp does not use a rail-to-rail structure. When the op-amp is powered by 5V, the common-mode input voltage (VCM) range is 0 to 3.7V. Be sure not to exceed this range when applying the input voltage.
4. The output voltage range of the op-amp is 0.3V to (VDD-0.3). Do not exceed the output range to prevent waveform distortion.
5. The slew rate (SR) of the op-amp is directly related to the load. When the load capacitance is 30pF, the slew rate can reach 5V/μs. With a load capacitance of 100pF, the output slew rate is 4V/μs. Choose the appropriate load capacitance based on the application scenario. (When selecting capacitance, consider the impact of parasitic capacitance.)
6. Since the built-in op-amp cannot output negative voltage, when using a reverse amplifier circuit, the input signal must be negative to ensure normal operation of the op-amp.
7. If the op-amp's offset voltage is adjusted using the corresponding offset voltage in the CONFIG register (which has been adjusted to the optimal value at the factory), simply set the OPnADJE register to a value other than 0xAA. (When using the adjustment bit for trimming, the external environment may not be optimal, which can result in suboptimal adjustment. Therefore, disable the adjustment bit and use the internal CONFIG settings.)
8. The input impedance of the OPA should not be too high, as this may cause abnormal functionality. It is recommended that the input impedance be less than 50K.

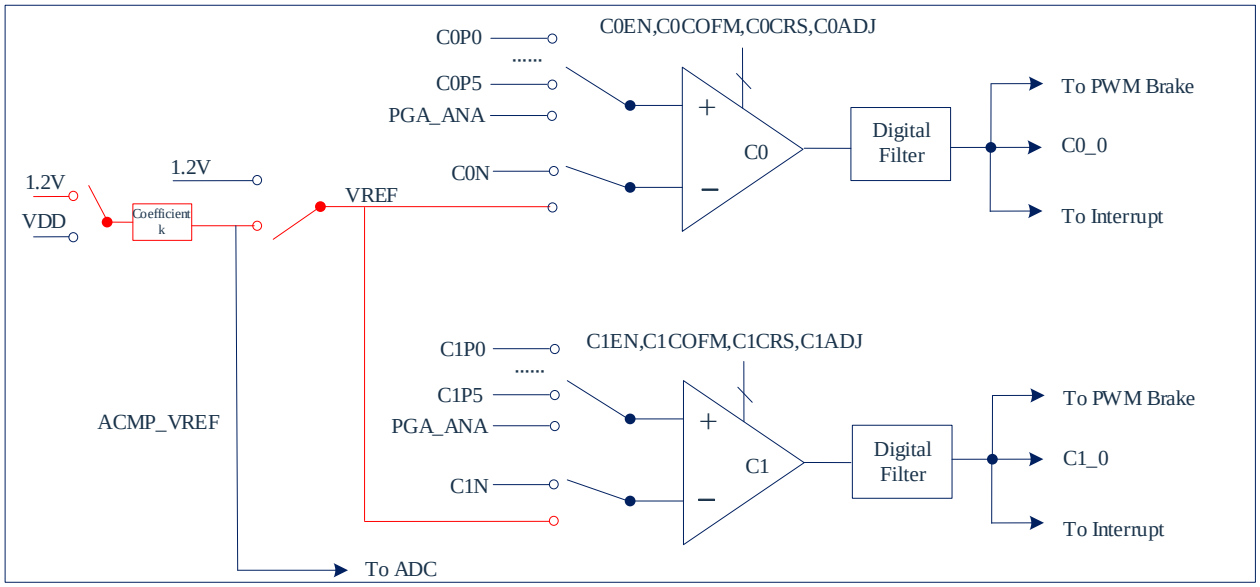
19. Analog Comparator

19.1 Overview

The built-in comparator is a general-purpose analog comparator integrated into the MCU. When the voltage at the positive input is greater than the voltage at the negative input, the comparator outputs a logical 1. Otherwise, it outputs a 0. The output polarity can also be changed via the output polarity select bit. Each comparator can generate an interrupt when its output value changes.

19.2 Block Diagram of the Built-in Comparator

The block diagram of the comparator is shown below:



19.3 Application Notes for the Built-in Comparator

1. Since the internal 1.2V reference voltage has limited drive capability, if the red path shown in the diagram is selected as the negative (“-”) input voltage for the comparator, it may cause the internal reference voltage to be pulled down. Therefore, it is not recommended to use this path as the negative (“-”) input voltage in applications where the reference voltage requirements are critical.

20. LVD

The CMS8S589x MCU has a built-in power supply detection function. When the Low Voltage Detection (LVD) module is enabled (LVDEN=1) and the voltage detection threshold (LVDSEL) is set, an interrupt will be triggered when the supply voltage drops below the set LVD threshold, alerting the user.

If the LVD module is enabled before entering sleep mode, the hardware for the module will not be turned off after entering sleep; it must be disabled via software (LVDEN=0).

When enabling the LVD module and its associated functions, it is important to follow the correct configuration sequence:

- When enabling the LVD module interrupt functionality, follow these steps:
  1) Configure the LVD voltage detection point (LVDSEL<2:0>);
  2) Enable the LVD module (LVDEN=1);
  3) Clear the LVD interrupt flag (LVDINTF=0);
  4) Enable the LVD interrupt (LVDINTE=1).
- When polling the LVD interrupt flag, follow these steps:
  1) Configure the LVD voltage detection point (LVDSEL<2:0>);
  2) Enable the LVD module (LVDEN=1);
  3) Clear the LVD interrupt flag (LVDINTF=0);
  4) Poll the LVD interrupt flag (LVDINTF).

Applicable Products: These application notes apply to the following product model: CMS8S589x.

21. LSE

21.1 Cautions

If the Watchdog Timer (WDT) is set to reset the system, and the LSE module is enabled, it is important to clear the watchdog counter promptly during the process of waiting for the LSE clock to stabilize. Otherwise, an undesired watchdog reset may occur, as shown in the figure below. Although the watchdog overflow time can be configured, it is typically in the millisecond range, which is shorter than the stabilization time of the LSE clock (approximately 1.5 seconds). If a timeout control is used, the timeout duration is also ⩾1.5 seconds. During the normal process of waiting for the LSE clock to stabilize, a millisecond-level watchdog overflow will cause an undesired watchdog reset.

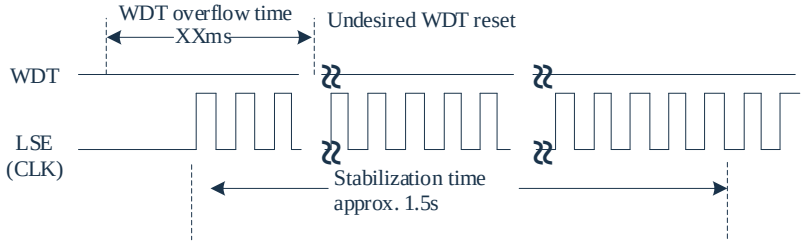
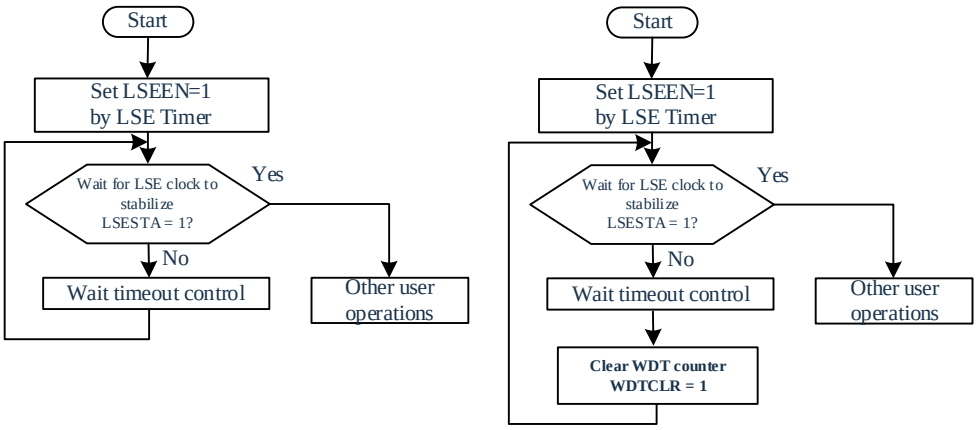


Figure 21-1: Watchdog reset during the LSE stabilization process

21.2 Recommended Operations

As shown in Figure 21-2 b, the recommended operation flow is to promptly clear the watchdog counter during the process of waiting for the LSE clock to stabilize.



a Common Operation Flow **b Recommended Operation Flow**

Figure 21-2: WDT operations during the LSE stabilization process

Recommended Programming Example: During the stabilization of the LSE clock, the watchdog counter should be cleared in a timely manner.

Common Example

```
void LSE_Config(void)
{
    /*(1)Enable LSE*/
    LSE_EnableLSE();
    /*(2) Wait for LSE stabilization
(approximately 1.5s)*/
    While( !LSE_GetLSEState())
    {
        ……//Timeout control
    }
    ……//Other operations
}
```

Recommended Example

```
void LSE_Config(void)
{
    /*(1)Enable LSE*/
    LSE_EnableLSE();
    /*(2) Wait for LSE stabilization
(approximately 1.5s)*/
    While( !LSE_GetLSEState())
    {
        ……//Timeout control
        WDT_ClearWDT();//Clear the watchdog
    }
    ……//Other operations
}
```

22. 1T Mode TA Register Operation Notes

22.1 Cautions

During TA register operations, no interruption is required between TA=0xAA and TA=0x55. Therefore, when operating on TA, interrupts must be disabled (EA=0). The common example is: EA = 0; → TA = 0xAA; → TA = 0x55;

When the instruction clock is set to ≥2T, this example works without any issues.

However, when the instruction clock is set to 1T, due to faster internal timing control, instructions must be prefetched. If an interrupt occurs while the kernel is executing EA = 0, it will first respond to the interrupt and complete the next instruction after EA = 0, before beginning the interrupt service routine. During this process, the discontinuity between TA=0xAA and TA=0x55 will cause the unlock register to fail.

22.2 Recommended Operations

Common Example	**Recommended Example**
EA = 0; TA =0xAA; TA =0x55;	EA = 0; nop (); TA =0xAA; TA =0x55;

23. Power On

During the power-on process, ensure that the chip's VDD and GND pins have good contact to guarantee the chip operates in the normal power-on mode. If the GND pin has poor contact and an IO pin is at a low input level, the power supply may deliver abnormal power to the chip through VDD and the IO (low level). This situation can potentially cause permanent damage to the chip's flash memory.

- During the programming process, ensure good contact between the VDD and GND pins of the chip to prevent power from being supplied to the chip via the programming interface, which could result in permanent damage to the flash memory.

Applicable Products: This application note is applicable to the following product models: CMS8S589x, CMS8S369x.

24. Sleep Mode (STOP)

After the chip enters the sleep mode (STOP), all circuits except the LVD and LSE modules are powered off (the LVD/LSE modules must be turned off via software). The system enters a low-power mode, and the digital circuits stop functioning.

When this mode is required for application, the LVR voltage threshold must be set to 1.8V. If the LVR voltage threshold is not correctly set, it may cause an abnormal reset of the chip during wake-up from sleep.

Applicable Products: This application note is applicable to the following product models: CMS80F262x, CMS80F261x.

25. Others

- For programs that require multiple consecutive statements to complete an operation, do not insert delays or breakpoints between the consecutive statements. If an interrupt occurs in the middle, it will interrupt the sequence of operations, which may affect the success rate of the operation.
- For bitwise operations on program registers, it is recommended to first read the value of the register into a temporary variable, then modify the value of the temporary variable according to the corresponding bits. Finally, write the value of the temporary variable back into the register.

26. Revision History

Version #	Date	Description of changes
V1.00	Dec. 2020	Initial release
V1.01	Feb. 2021	Added notes for using external oscillators.
V1.02	Apr. 2021	Added a diagram illustrating the time intervals for FLASH operations.
V1.03	Nov. 2021	Added a section on FLASH operation notes and included some notes on ADC.
V1.04	Dec. 2021	Added a detailed description of the TA write timing.
V1.05	Jun. 2022	Added explanations on LSE wake-up and IO ports (serial port, I2C) pull-up, open-drain, as well as FLASH operation.
V1.06	Sep. 2022	1) Added a new section 18 on op-amp offset voltage. 2) Section 4 ADC: Added note 6). 3) Added a new section 19 LVD. 4) Added a new section 20 LSE.
V1.07	Nov. 2022	Modified section 3 regarding IO configuration.
V1.0.8	Jan. 2023	Added section 21 on TA register operation notes in 1T mode.
V1.0.9	May 2023	1) Updated section 7 on the applicable chip types for USART. 2) Added a new section 22 on power-up precautions.
V1.0.10	Jun. 2023	1) Added section 14.3 on FLASH CRC precautions.
V1.0.11	Sept. 2023	1) Updated section 18 on op-amp precautions. 2) Added application notes for comparators.
V1.0.12	Jan. 2024	Added a new section 24 on sleep mode (STOP) precautions.
V1.1.0	Mar. 2025	Corrected errors in section 19.1 and Chapter 24.