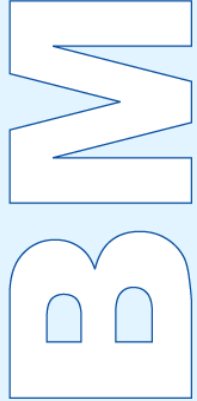




ATOM Family

BM-ATOM1.1-V1.2



Brief Manual of ATOM1.1 Family

4-bit Microcontrollers with Reduced 8051 Architecture

V1.2

MAY 2010

- ◆ CORERIVER Semiconductor reserves the right to make corrections, modifications, enhancements, improvements, and other changes to its products and services at any time.
- ◆ CORERIVER shall give customers at least a three month advance notice of intended discontinuation of a product or a service through its homepage.
- ◆ Customers should obtain the latest relevant information before placing orders and should verify that such information is current and complete.
- ◆ The CORERIVER products listed in this document are intended for usage in general electronics applications. These CORERIVER products are neither intended nor warranted for usage in equipment that requires extraordinarily high quality and/or reliability or a malfunction or failure of which may cause loss of human life or bodily injury.

1. Product Overview

2. Features

3. Block Diagram

4. Pin Configurations

5. Pin Descriptions

6. Function Descriptions

✓ CPU Descriptions

- Memory Organization
- SFR Map and Description
- Instruction Set Summary
- CPU Timing

✓ Peripheral Descriptions

- I/O Ports
- Clock Configuration
- Carrier Frequency Generation
- LVD (Low Voltage Detector)
- WDT (Watchdog Timer)
- Reset Circuit
- Power Management
- IAP (In Application Programming)

7. Absolute Maximum Ratings

8. DC Characteristics

9. AC Characteristics

10. Package Dimensions

11. Product Numbering System

12. Supporting Tools

13. Appendix

A. Instruction Set

B. SFR Descriptions

C. Update History

1. Product Overview

Preliminary

◆ ATOM1.1 Family - GC49C501G1 Series (Low Cost, Low Power Application MCU)

Product	Mask-ROM (byte)	FLASH (byte)	EEPROM (byte)	RAM (Nibble)	Volt (V)	Freq. (MHz)	T/C (16bits)	Serial I/O	WDT	REM Output	IR. LED Drive Tr.	I/O Pins	Package	Others	Available Time
GC49C501G1-SO24I	-	1K	(128)	64	1.8~5.5	10 (5)	-	-	1	1	Yes	18 (20)	24-SOIC	POR/LVD Ring OSC ISP/IAP	NOW
GC49C501G1-SJ20I	-	1K	(128)	64	1.8~5.5	10 (5)	-	-	1	1	Yes	14 (16)	20-SOIC (JEDEC)	POR/LVD Ring OSC ISP/IAP	NOW
GC41C501G1-SO24I	1K	-	-	64	1.8~5.5	10 (5)	-	-	1	1	Yes	18 (20)	24-SOIC	POR/LVD Ring OSC	NOW
GC41C501G1-SJ20I	1K	-	-	64	1.8~5.5	10 (5)	-	-	1	1	Yes	14 (16)	20-SOIC (JEDEC)	POR/LVD Ring OSC	NOW

* User may use part of program area (128 bytes) as EEPROM, which can be modified by IAP function during S/W operation.

* Max. operating frequency of ATOM1.1 family is 5 MHz when VDD is less than 2.7 V.

2. Features

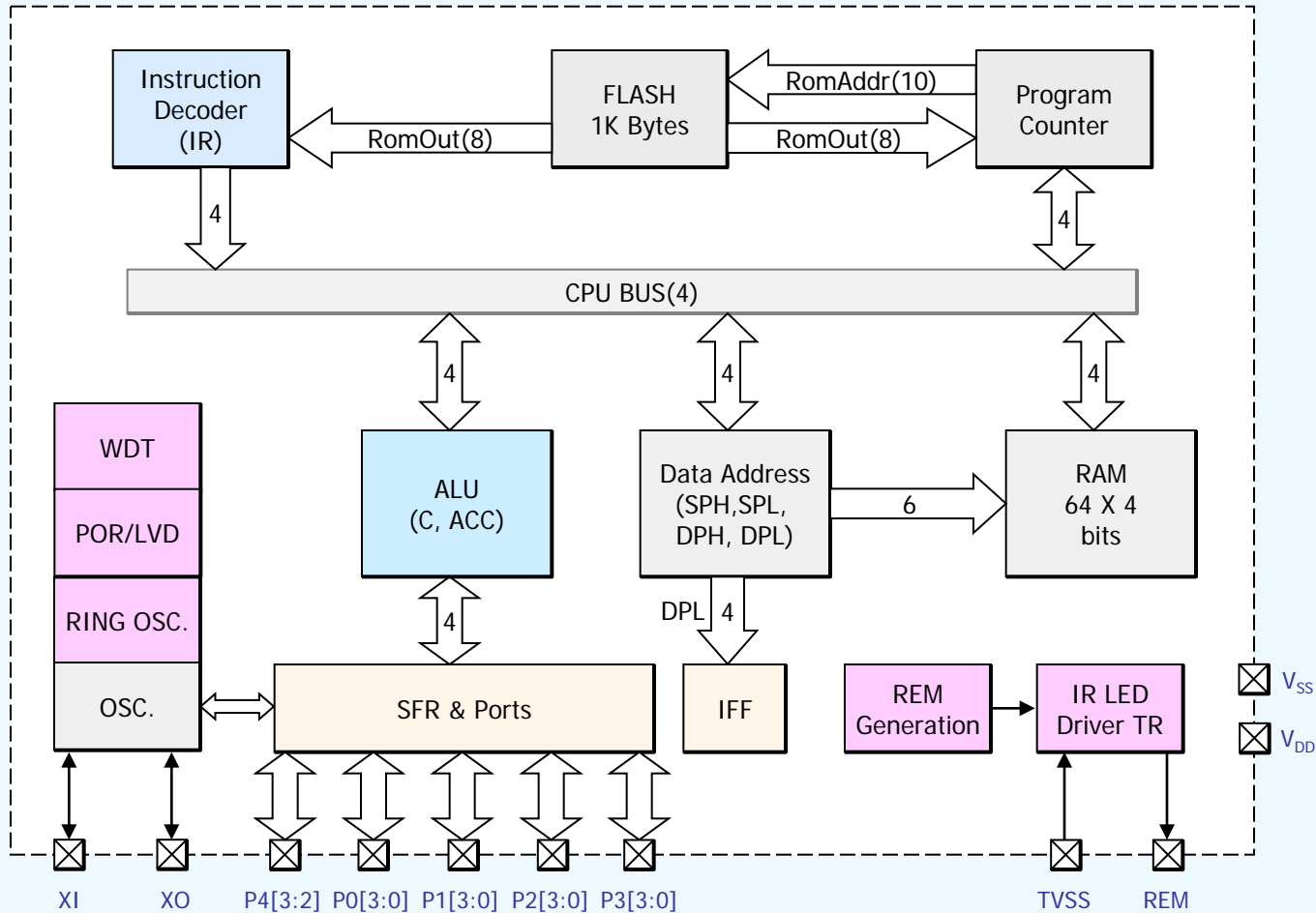
- ◆ CPU
 - ✓ 4-bit reduced 8051 architecture
 - ✓ Continuous program addressing, not paged.
 - ✓ **51** instructions including push, pop and logic inst.
 - ✓ Instruction cycle : $F_{\text{SYS}}/6$
 - ✓ Multi-level subroutine nesting with RAM based stack.
- ◆ On-chip Memories
 - ✓ FLASH : 1024 bytes (including 128 EEPROM)
 - ✓ RAM : 64 nibbles (including stack)
- ◆ ISP (In System Programming) of FLASH
- ◆ IAP (In Application Programming) of FLASH
- ◆ I/O Ports
 - ✓ P0 : 4-bit parallel I/O (Open drain output)
 - ✓ P1 : Parallel I/O (Open drain output)
4-bit for 24-pin, 2-bit for 20-pin.
 - ✓ P2, P3 : 4-bit parallel/bit-selectable I/O (Open drain output)
 - ✓ P4 : 2-bit Parallel I/O (Open drain output).
for 24-pin packages.
- ◆ REM output (Remote control transmitter)
 - ✓ Built-in Transistor for I.R. LED Drive
 - ✓ $I_{\text{OL}} = 300 \text{ mA (Max.)}$ at $V_{\text{DD}} = 3\text{V}$ and $V_{\text{O}} = 0.4\text{V}$
- ◆ Carrier Pulse Generation : 7 types
- ◆ Built-in Oscillator
 - ✓ Crystal/Ceramic resonator
 - ✓ Internal oscillator : 8MHz
- ◆ Built-in Reset
 - ✓ Power-on Reset, Power-fail Reset
 - ✓ WDT (Watch-Dog Timer) Reset
 - ✓ Clock switching reset
- ◆ Power Management
 - ✓ Power-down (stop) mode
 - ✓ Release stop by input changes
 - ✓ Sleep mode

2. Features

- ◆ Power Consumption
 - ✓ Stop mode : <math>< 0.1\mu\text{A}</math> (Typ.) at 2.0V
1 μA (Max.) at 5.0V
 - ✓ Normal mode : 400 μA (Typ.) at 2.0V, $F_{\text{SYS}} = 4\text{ MHz}$
- ◆ Operating frequency vs. voltage
 - ✓ Max. $F_{\text{OSC}} = 10\text{ MHz}$ ($2.7\text{ V} \leq V_{\text{DD}} \leq 5.5\text{V}$)
 - ✓ Max. $F_{\text{OSC}} = 5\text{ MHz}$ ($1.8\text{ V} \leq V_{\text{DD}} < 2.7\text{V}$)
- ◆ Operating temperature : -40 °C ~ 85 °C
- ◆ ESD protection
 - ✓ HBM : 2,000V (JESD22-A114E)
 - ✓ MM : 200V (JESD22-A115-A)
 - ✓ CDM : 800V (JESD22-C101-C)
- ◆ Latch-up protection up to $\pm 200\text{mA}$
- ◆ Package
 - ✓ 24-pin SOIC
 - ✓ 20-pin SOIC

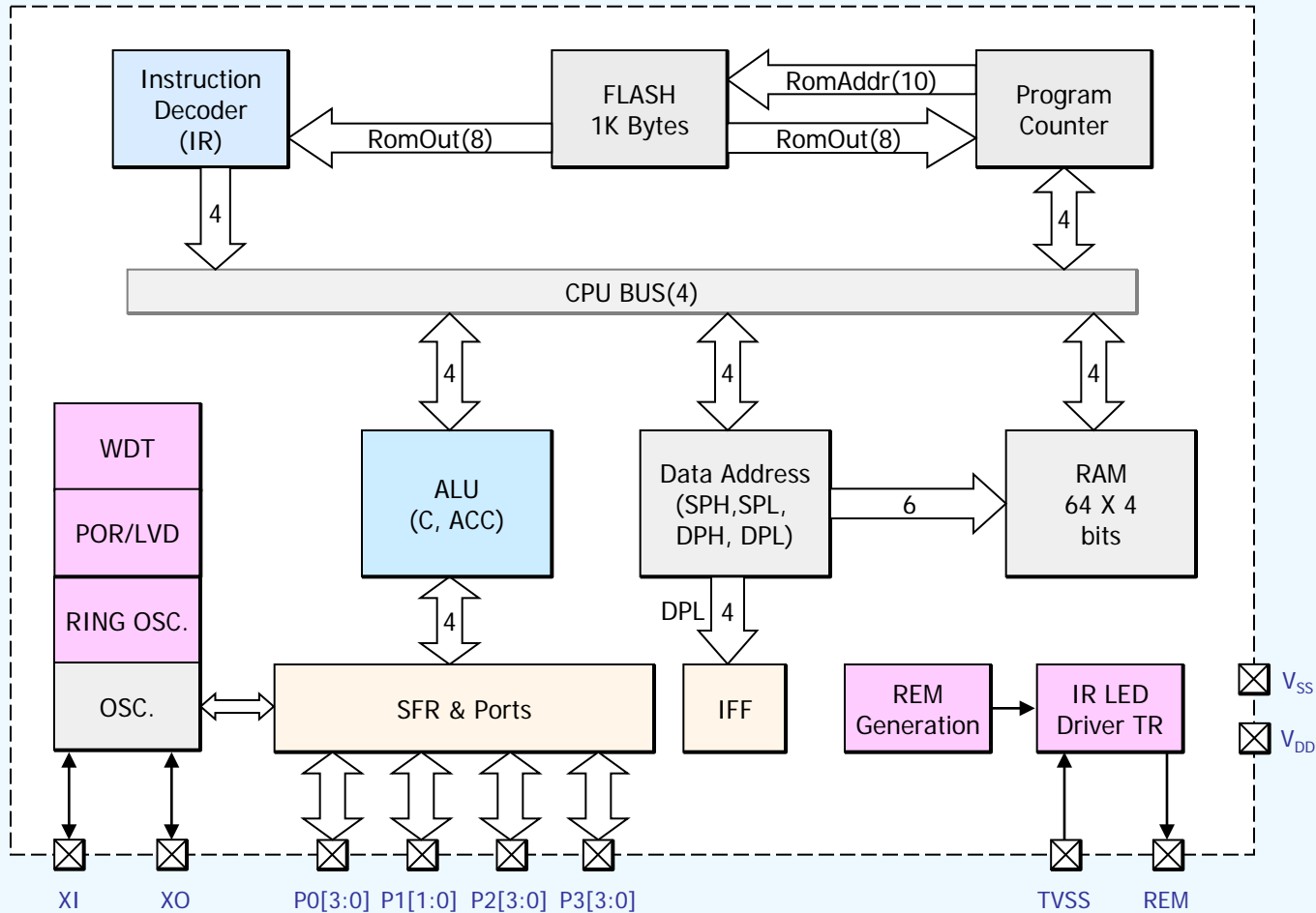
3. Block Diagram (24-PIN)

Preliminary

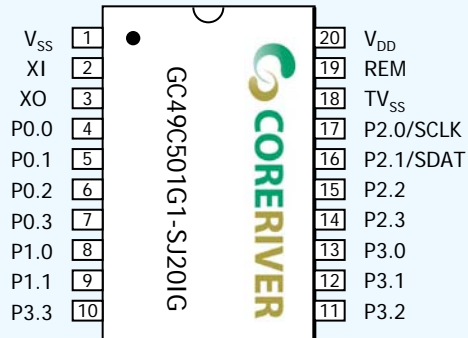


3. Block Diagram (20-PIN)

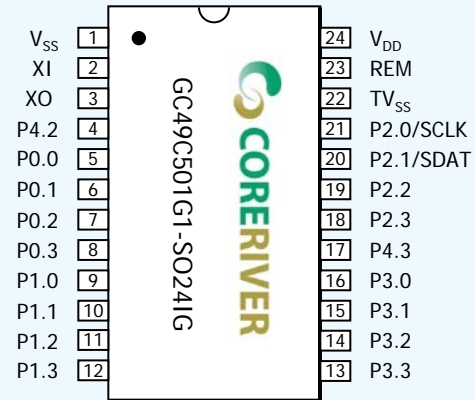
Preliminary



4. Pin Configurations



[20-SOIC]



[24-SOIC]

5. Pin Description (20-pin/24-pin)

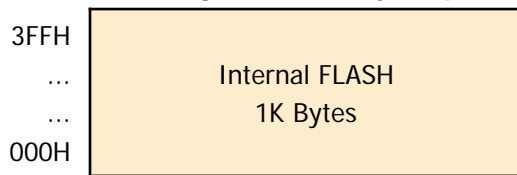
Symbol	Direction	Description	Remark
V _{DD}	Power	Power Supply	
V _{SS}	Power	Ground	
REM	Output	Output for IR LED drive Transistor. The transistor is n-channel device.	
TV _{SS}	Power	Ground for IR LED drive Transistor	
XI	Input	Input to the inverting oscillator amplifier.	
XO	Output	Output from the inverting oscillator amplifier.	
P4[3:2]	Input/Output	Parallel Input/Output port (Only for 24-pin packages) Each bit can be individually set or cleared. Schmitt Trigger input and open-drain output with internal pull-up TR.	
P0[3:0]	Input/Output	Parallel Input/Output port. Schmitt Trigger input and open-drain output with internal pull-up TR. The STOP mode is released by "L" input of each pin.	
P1[1:0]	Input/Output	Parallel Input/Output port. Schmitt Trigger input and open-drain output with internal pull-up TR. The STOP mode is released by "L" input of each pin.	
P1[3:2]	Input/Output	Parallel Input/Output port (Only for 24-pin packages) Schmitt Trigger input and open-drain output with internal pull-up TR. The STOP mode is released by "L" input of each pin.	
P2[3:0]	Input/Output	Parallel Input/Output port. Each bit can be individually set or cleared. Schmitt Trigger input and open-drain output with internal pull-up TR. P2 can be configured as a push-pull output port. P2[0] and P2[1] are also used for ISP of FLASH memory.	
P3[3:0]	Input/Output	Parallel Input/Output port. Each bit can be individually set or cleared. Schmitt Trigger input and open-drain output with internal pull-up TR.	

6.1. Memory Organization

◆ Address Space

- ✓ Program memory : 1K Bytes.
Continuously addressed by Byte.
- ✓ Indirect data memory : 64 Nibbles.
Bit accessible.
- ✓ Special function registers : 16 Registers.
Directly addressed.
- ✓ Indirect function flags : 16 bits.
Bit position is selected by DPL.

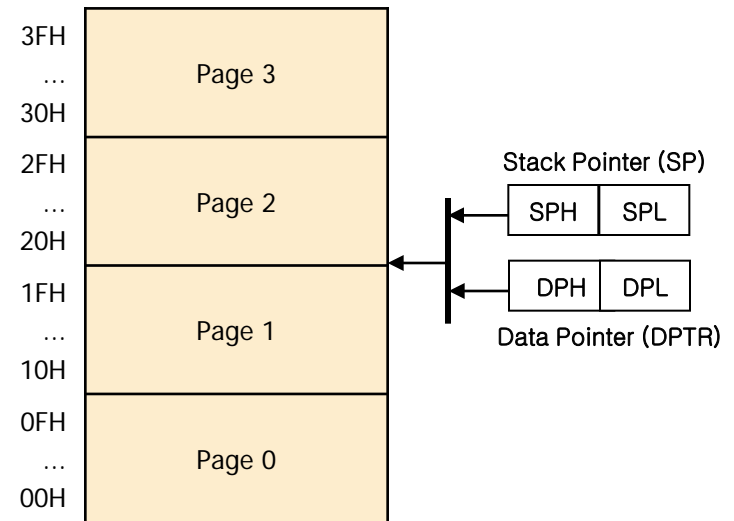
[Program Memory Map]



[Special Function Register Map]

0CH	P3	CKCFG	IOCFG	-
08H	P2	IAPCON	GDL	GDH
04H	P1	REMC	SPL	SPH
00H	P0	P4	DPL	DPH

[RAM Map]



[Indirect Function Flag Map]

15	14	13	12	11	10	9	8
STOP	SLEEP	WDTE	WDTR	MAP1	MAP0	P4.2	P4.3
7	6	5	4	3	2	1	0
P3.3	P3.2	P3.1	P3.0	P2.3	P2.2	P2.1	P2.0

6.2. SFR Brief Description

Register	Address	Description	Power-On Reset Value	Other Reset Value
P0	00H	Port 0 output register.	1111	1111
P4	01H	Port 4 output register.	1111	1111
DPL	02H	The low nibble of data pointer (DPTR).	0000	0000
DPH	03H	The high nibble of data pointer (DPTR).	--00	--00
P1	04H	Port 1 output register.	1111	1111
REMC	05H	REM output control register.	0000	0000
SPL	06H	The low nibble of stack pointer (SP).	1111	1111
SPH	07H	The high nibble of stack pointer (SP).	--01	--01
P2	08H	Port 2 output register.	1111	1111
IAPCON	09H	IAP (In Application Programming) Control register. Can be accessed only if MAP1 is set and MAP0 is cleared.	0000	0000
GDL	0AH	The low nibble of general purpose data register	0000	0000
GDH	0BH	The high nibble of general purpose data register	0000	0000
P3	0CH	Port 3 output register.	1111	1111
CKCFG	0DH	The clock configuration register. Initialized only by power-on-reset.	0000	uuuu
IOCFG	0EH	The I/O port configuration register. Initialized only by power-on-reset.	0000	uu0u
-	0FH	Reserved	----	----

- : Unimplemented bit. Read as 0.

u: Remains unchanged.

6.2. Indirect Function Flag (IFF) Description

◆ Indirect Function Flag (IFF)

- ✓ Write only, access using the instructions: MOV L, #n, SETB @L, CLR @L
- ✓ The individual set/clear of ports is available only if the package type supports corresponding parallel port.

Flag	Address (DPL)	Description	Reset Value
STOP	15	Enter stop mode. Not set until all pins of P0 and P1 are high.	0
SLEEP	14	Enter sleep mode. Released by WDT reset.	0
WDTE	13	Enable flag of WDT. If this flag is cleared, WDT stops running and holds the state. This flag can be modified if and only if MAP1 bit is set and MAP0 bit is cleared. This flag is also set by H/W when user sets SLEEP flag or writes IAPCON SFR.	1
WDTR	12	Reset Watch Dog Timer. Set by S/W. Cleared by H/W after WDT is reset.	0
MAP1	11	Address map extension bit 1 for SFR/IFF.	0
MAP0	10	Address map extension bit 0 for SFR/IFF. Do not set this flag for the future compatibility.	0
P4.2	9	Individual bit set/clear for P4	1
P4.3	8	Individual bit set/clear for P4	1
P3.3	7	Individual bit set/clear for P3	1
P3.2	6	Individual bit set/clear for P3	1
P3.1	5	Individual bit set/clear for P3	1
P3.0	4	Individual bit set/clear for P3	1
P2.3	3	Individual bit set/clear for P2	1
P2.2	2	Individual bit set/clear for P2	1
P2.1	1	Individual bit set/clear for P2	1
P2.0	0	Individual bit set/clear for P2	1

6.3. Instruction Set Summary (1/2)

Preliminary

- ◆ Refer to Appendix A (Instruction Set) for more details.

Type	Instruction	Description
Arithmetic	ADD A, #data INC A DEC A ADD A, @DP ADDC A, @DP SUB A, @DP INC @DP DEC @DP	Add data to ACC. Increment ACC. Decrement ACC. Add the indirect memory nibble to ACC. Add the indirect memory nibble to ACC with the Carry in C. Subtract the indirect memory nibble from ACC. Increment the indirect memory nibble. Decrement the indirect memory nibble.
Logical	CLR A CPL A RRC A ANL A, @DP ORL A, @DP XRL A, @DP	Clear ACC. Complement ACC. Rotate right ACC with Carry flag. Logical AND for ACC and the indirect memory nibble. Logical OR for ACC and the indirect memory nibble. Logical Exclusive-OR for ACC and the indirect memory nibble.
Data Transfer	MOV dir, A MOV A, dir MOV A, @DP MOV A, #data MOV L, @DP MOV @DP, A MOVI @DP, A MOVD @DP, A XCH A, @DP MOVI @DP, #data MOV L, #data MOV H, #data PUSH A POP A	Move ACC to the special function register. Move the special function register to ACC. Move the indirect memory nibble to ACC. Move data to ACC. Move the indirect memory nibble to DPL. Move ACC to the indirect memory nibble. Move ACC to the indirect memory nibble and increment the data pointer (DPH,DPL). Move ACC to the indirect memory nibble and decrement the data pointer (DPH,DPL). Exchange ACC and the indirect memory nibble. Move data to the indirect memory nibble and increment the data pointer (DPH,DPL). Move data to DPL. Move data to DPH. Push ACC to stack. Pop stack to ACC.

6.3. Instruction Set Summary (2/2)

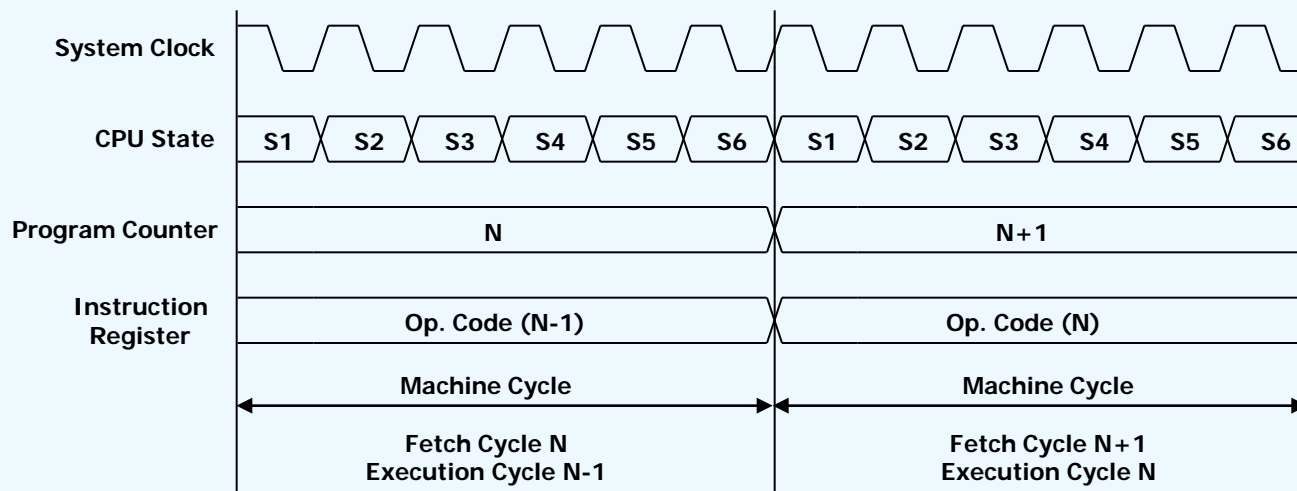
- ◆ Refer to Appendix A (Instruction Set) for more details.

Type	Instruction	Description
Branch	CJNE @DP, #data, rel CJNE L, #data, rel CJNE A, dir, rel CJNE A, @DP, rel CJLE A, @DP, rel CJNE A, #data, rel DJNZ A, rel JB bit, rel JNB bit, rel JC rel JNC rel JMP addr CALL addr RET NOP	Jump if the indirect memory nibble is not equal to the data. Jump if DPL is not equal to the data. Jump if ACC is not equal to the special function register. Jump if ACC is not equal to the indirect memory nibble. Jump if ACC is less than or equal to the indirect memory nibble. Jump if ACC is not equal to the data. Decrement ACC. Jump if the result is not zero. Jump if the indirect memory bit is 1. Jump if the indirect memory bit is 0. Jump if C is 1. Jump if C is 0. Jump to given address. Call subroutine. Return from subroutine. No operation.
Bit & Misc.	SETB @L CLR @L SETB bit CLR bit SETB C CLR C INC DPTR DEC DPTR	Set the indirect function flag. Clear the indirect function flag. Set the indirect memory bit. Clear the indirect memory bit. Set Carry flag. Clear Carry flag. Increment the data pointer. Decrement the data pointer.

6.4. CPU Timing

Preliminary

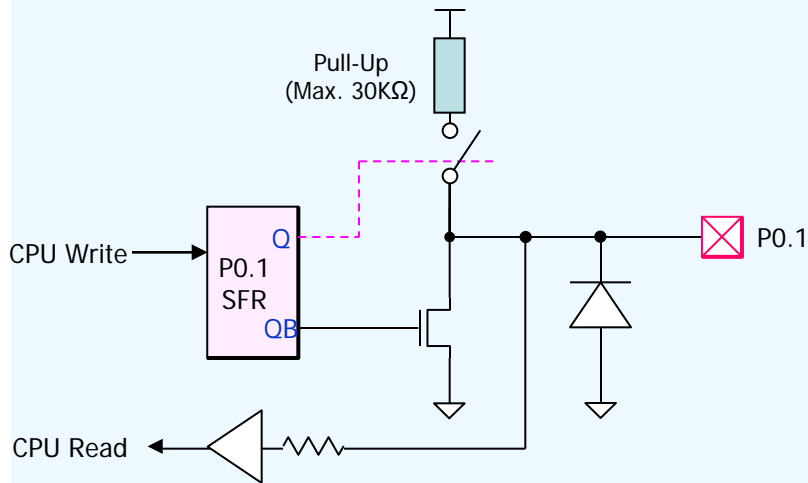
- ◆ CPU takes 6 clocks for a machine cycle.
- ◆ Any instruction except branch instructions completes in one machine cycle.
- ◆ All branch instruction consumes 2 machine cycles whether the branch is taken or not.
- ◆ The state of SFR, I/O ports, or IFF flags changes at the end of an instruction (S6).



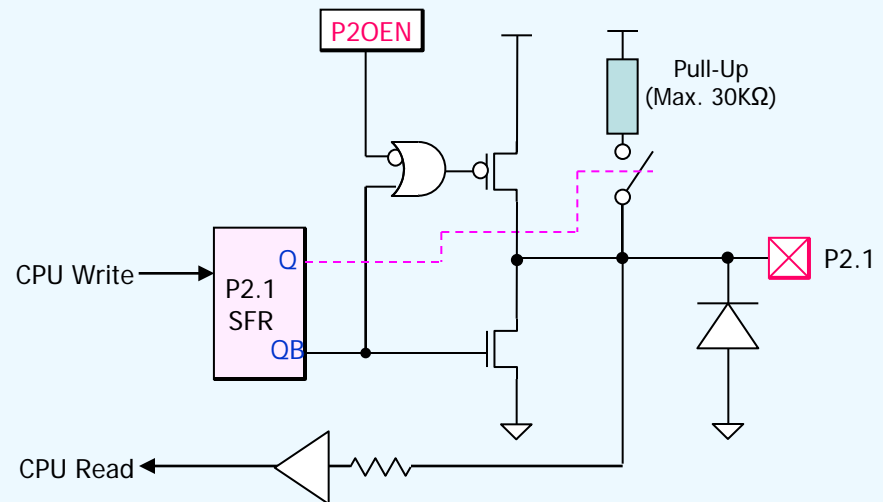
6.5. I/O Ports : PORT0 ~ PORT4

Preliminary

- ◆ All ports are initialized asynchronously on power-up.
- ◆ Pull-up enable and input by default (reset).
- ◆ Open drain active low output.
- ◆ P2[3:0] may be configured as push-pull output port.
- ◆ CPU always write to SFR register, but reads port pin.
- ◆ Retains the previous state in stop mode or sleep mode.



Circuit of P0[3:0], P1[3:0], P3[3:0], P4[3:2]



Circuit of P2[3:0]

6.5. I/O Ports : Mapping

◆ IOCFG

- ✓ This SFR is initialized to default state only by power-on-reset. Only the P2OEN bit is cleared by other resets.

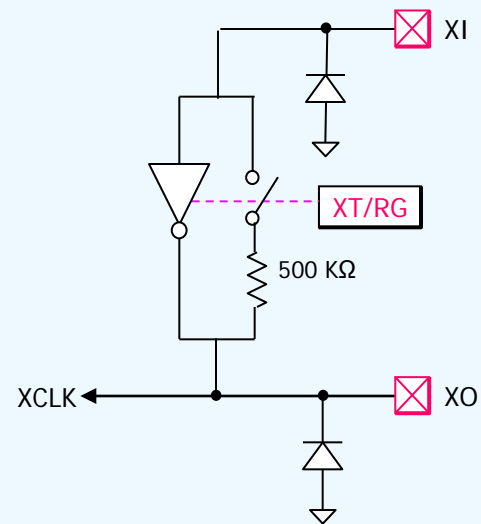
✓ IOCFG (0Eh) : I/O Port Configuration Register

IOMAP1	IOMAP0	P2OEN	-
R/W(0)	R/W(0)	R/W(0)	R/W(0)

- P2OEN : Configure P2 as a push-pull output port .
- IOMAP[1:0] : Configure I/O ports mapping .

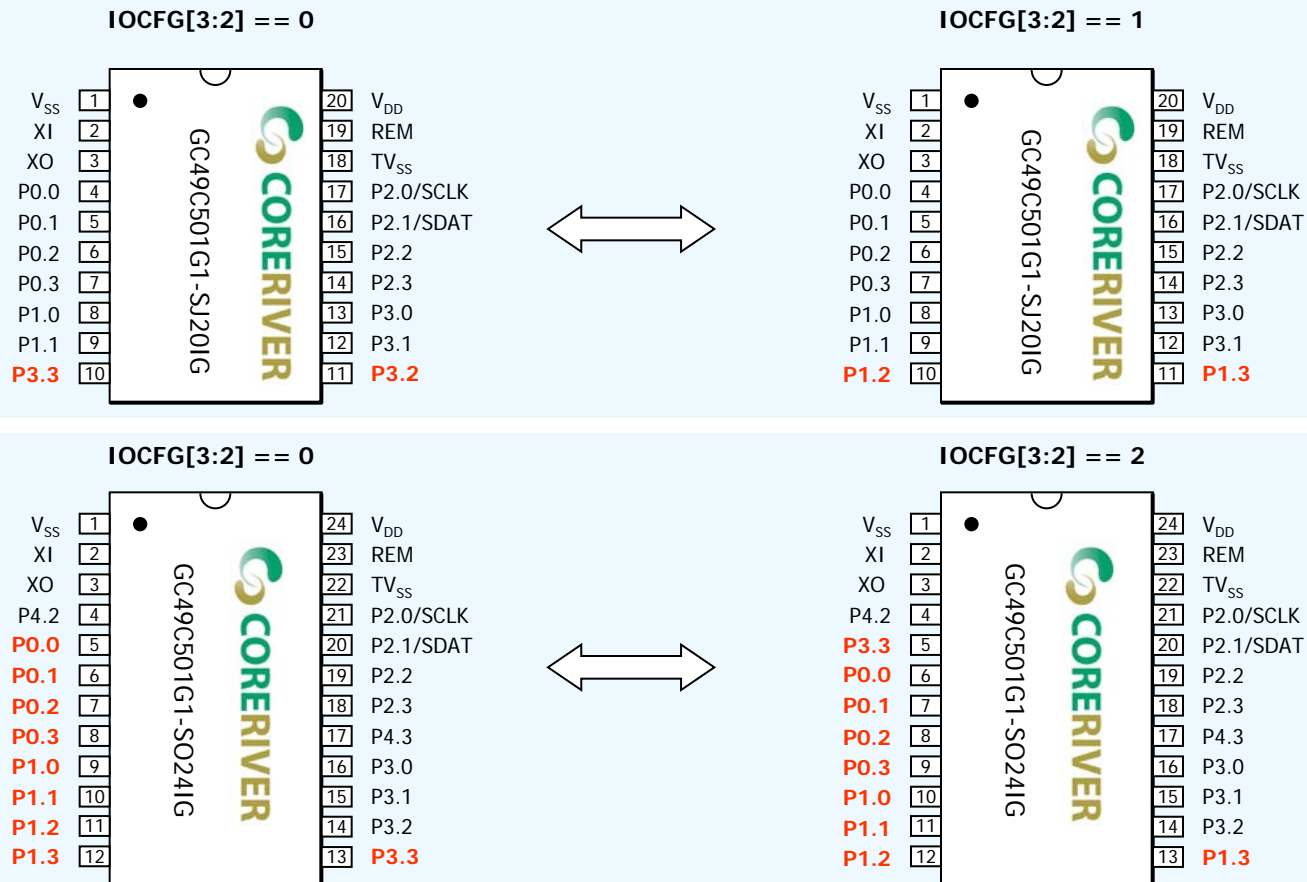
IOMAP 1	IOMAP0	Ports Mapping
0	0	Default.
0	1	Optional 20-pin I/O Port Mapping
1	0	Optional 24-pin I/O Port Mapping
1	1	Reserved

◆ XI/XO



6.5. I/O Ports : I/O Mapping

- ◆ User may select I/O port mapping by setting IOCFG SFR.
- ◆ The functionality of each I/O pins is the same for any mapping.
- ◆ This configuration option is useful when the pin-to-pin compatibility with existing devices is essential.



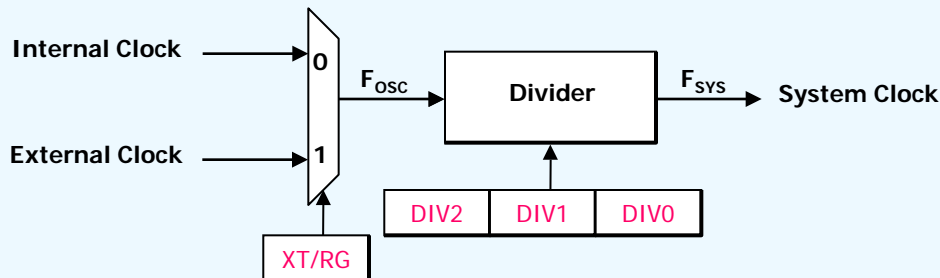
6.6. Clock Configuration

- ◆ Two System Clock Sources : Internal Ring OSC. or External Resonator/Crystal
- ◆ Default System Clock is Ring OSC.
- ◆ When user changes the clock source (XT/RG bit), internal reset is generated.
- ◆ Internal reset does not affect CKCFG.
- ◆ The configuration SFR (CKCFG) is initialized by power-on reset.
- ◆ User may change clock frequency during operation by changing divide option.

✓ **CKCFG (0Dh)** : The clock configuration register.

XT/RG	DIV2	DIV1	DIV0
R/W(0)	R/W(0)	R/W(0)	R/W(0)

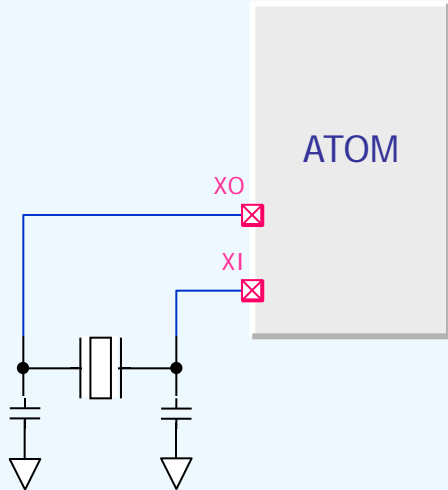
- **XT/RG** : System clock source selection.
 - 0 = Internal Ring oscillator is selected as system clock.
External clock osc. is disabled.
 - 1 = External clock is selected as system clock.
Internal Ring oscillator is disabled.
Do not set this bit for 8-pin devices.
- **DIV[2:0]** : System clock divider selection.



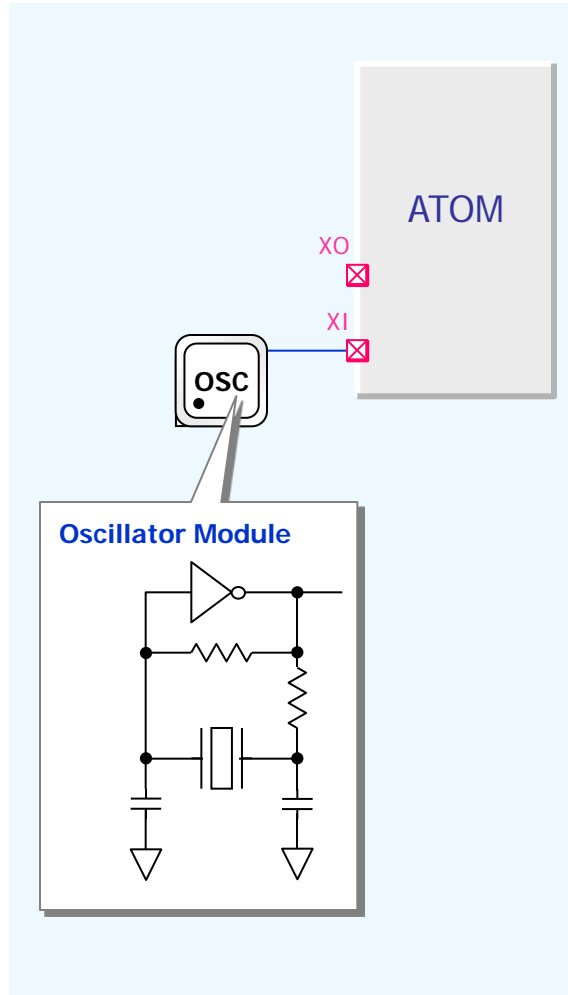
DIV2	DIV1	DIV0	F _{sys}
0	0	0	F _{osc}
0	0	1	F _{osc} /2
0	1	0	F _{osc} /4
0	1	1	F _{osc} /8
1	0	0	F _{osc} /16
1	0	1	F _{osc} /32
1	1	0	F _{osc} /64
1	1	1	-

6.6. Clock Configuration: Guideline

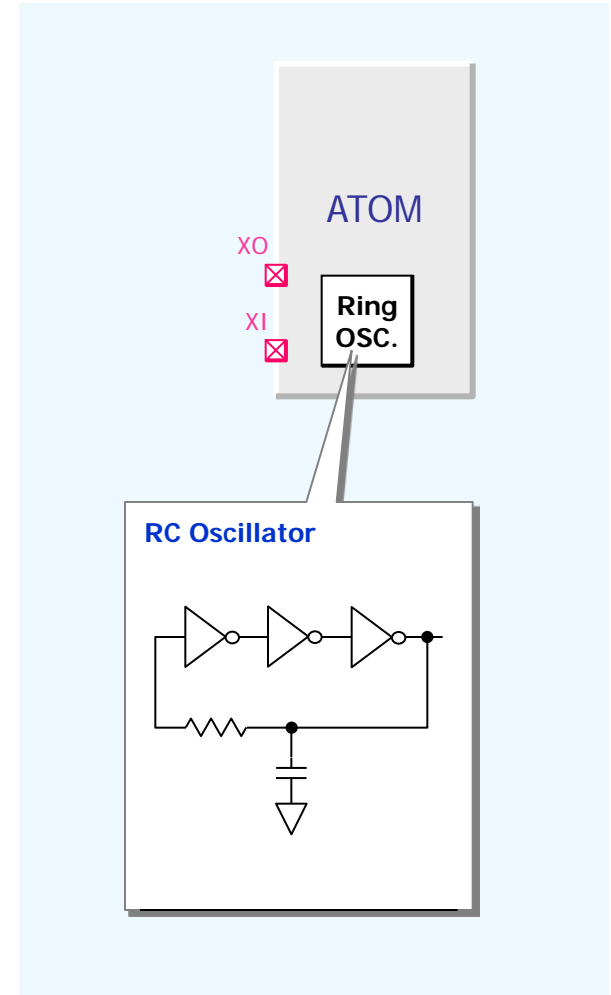
◆ Resonator / Crystal Oscillator



◆ Oscillator Module



◆ Internal Ring Oscillator



6.7. Carrier Frequency Generation

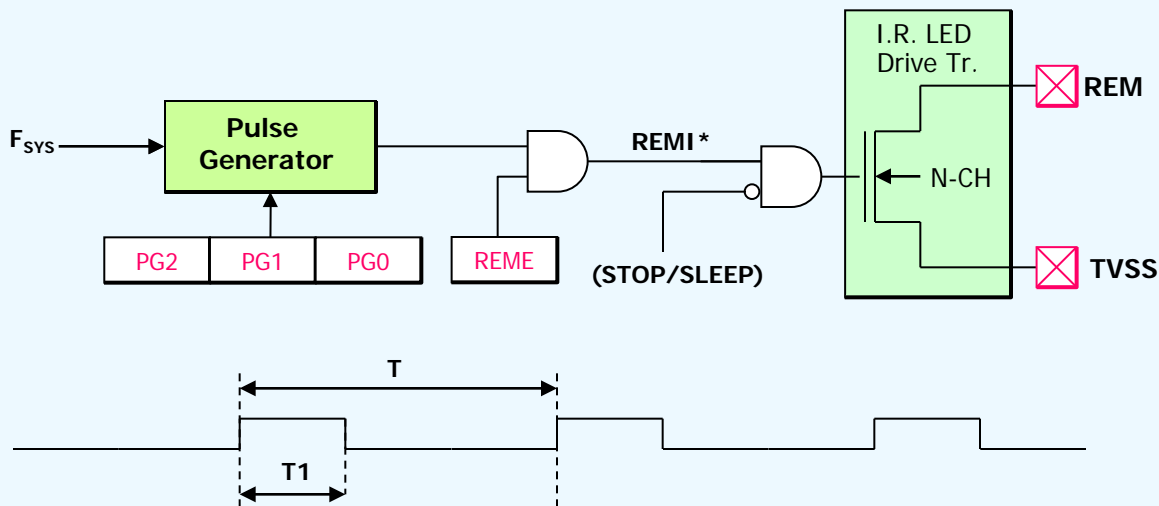
◆ Support 7 types of carrier frequency.

✓ **REMC** (05h) : The REM Output Control Register.

REME	PG2	PG1	PG0
R/W(0)	R/W(0)	R/W(0)	R/W(0)

- PG[2:0] : Carrier Frequency Selection.
- REME : REM Output Enable.

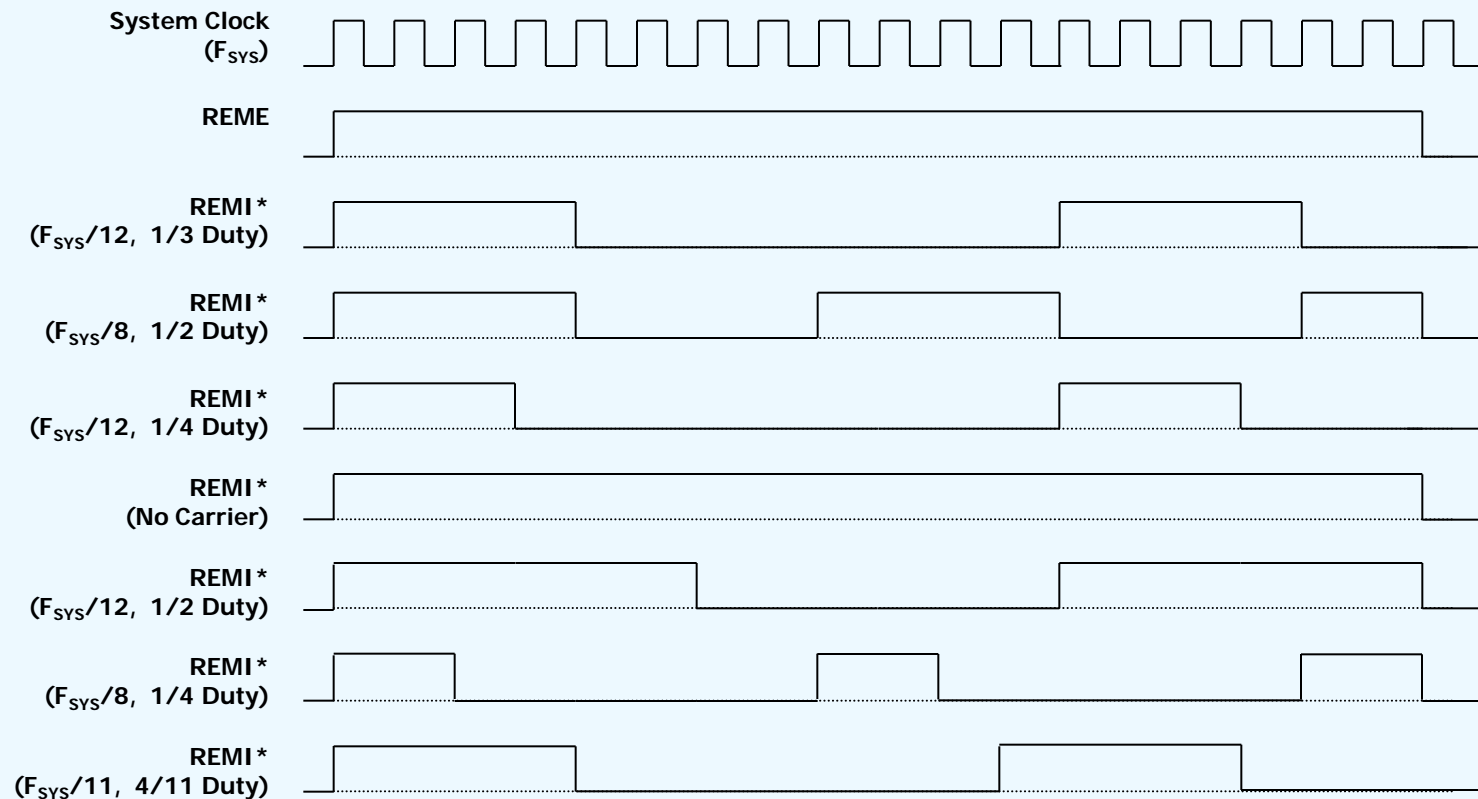
REME	PG2	PG1	PG0	Transmission Control (REMI)
0	X	X	X	0 (Disable)
1	0	0	0	$1/T = F_{SYS}/12, T1/T = 1/3$
1	0	0	1	$1/T = F_{SYS}/8, T1/T = 1/2$
1	0	1	0	$1/T = F_{SYS}/12, T1/T = 1/4$
1	0	1	1	1 (No Carrier)
1	1	0	0	$1/T = F_{SYS}/12, T1/T = 1/2$
1	1	0	1	$1/T = F_{SYS}/8, T1/T = 1/4$
1	1	1	0	$1/T = F_{SYS}/11, T1/T = 4/11$
1	1	1	1	1 (No Carrier)



6.7. Carrier Frequency Generation

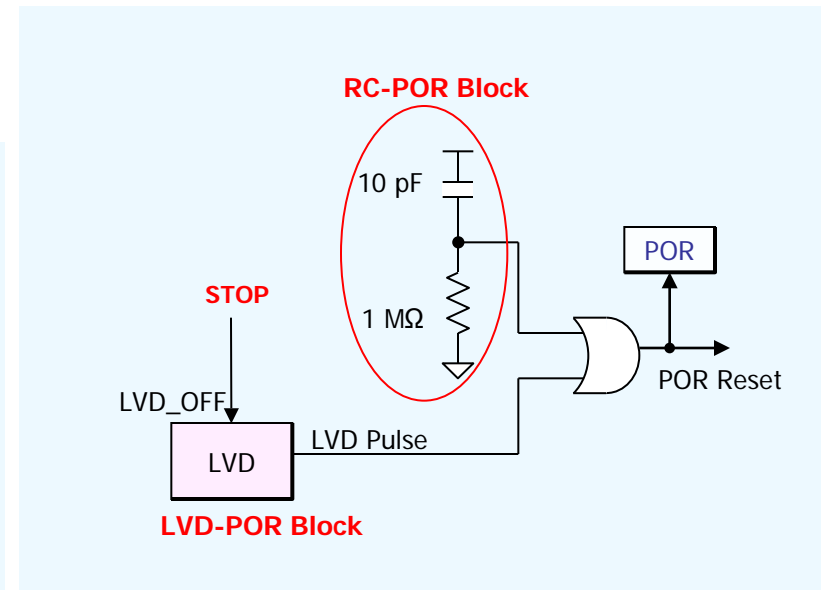
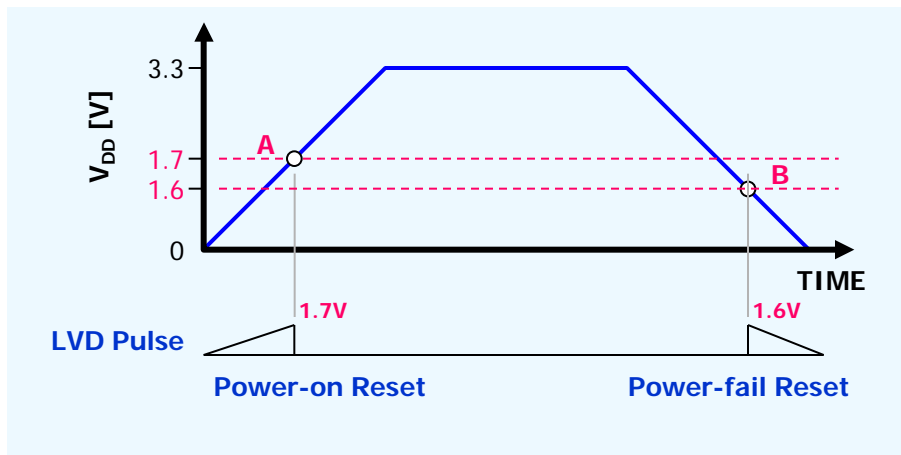
◆ Waveform Example

- ✓ REM output is the inverse of REMI*
- ✓ Since the IR. LED drive transistor in ATOM is a N-Type, IR. LED is turned on when REMI* is high.



6.8. POR & LVD : Power-On Reset

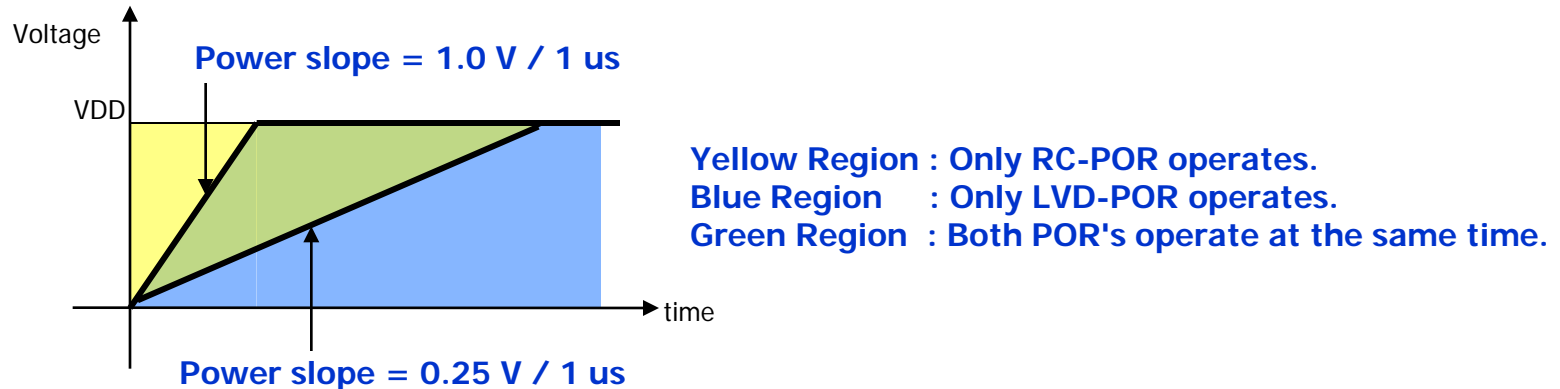
- ◆ On-chip power-on reset is a logical OR of RC-POR and LVD-POR
- ◆ RC-POR operates when the rising time of power (V_{DD}) is short.
- ◆ On-chip LVD
 - ✓ Provides power-on reset when the rising time of power is relatively long.
 - ✓ Power-on reset voltage is 1.7 V.
 - ✓ Provides power-fail reset when the power goes down below 1.6 V.
- ◆ After POR pulse is off, the internal clock stabilization counter starts to run, which lengthens power-on reset about 4.5 ms.



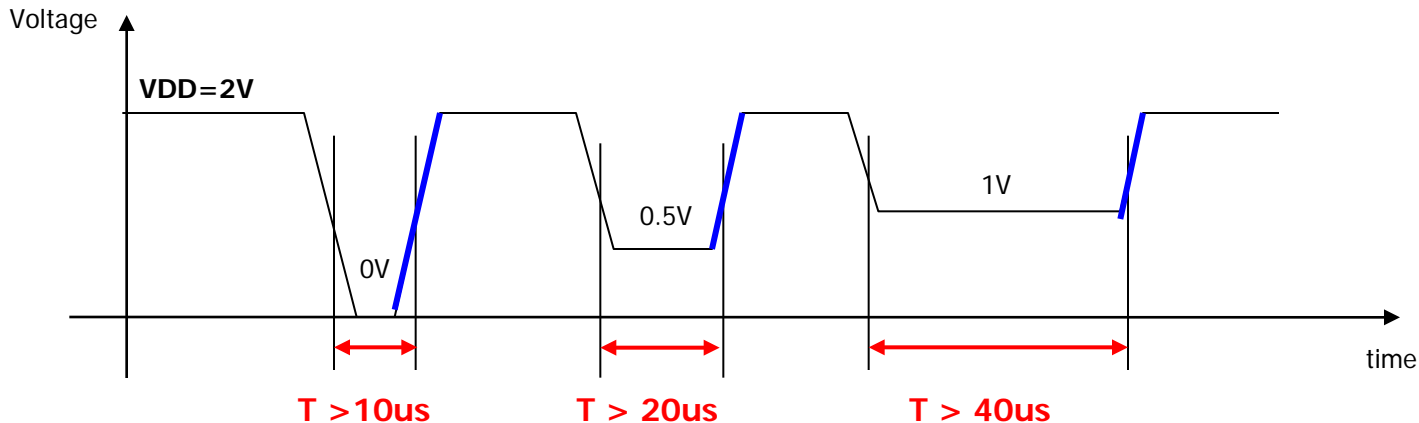
6.8. POR & LVD : Condition for power notch

Preliminary

- ◆ Power-on-reset is independent of power-rising slope.



- ◆ The cases of reset generation by VDD notch



When VDD fails for a short time, the duration of notch (T) has limitation like above for the successful POR operation.

The duration (T) will be changed by the VDD value and the transition time

6.9. WDT (Watchdog Timer)

Preliminary

◆ WDT

- ✓ Free running counter which resets CPU every 2^{17} system clock cycles.
- ✓ Although the counter length is fixed, WDT overflow period may vary according to the current frequency of system clock.
- ✓ WDT is halt in STOP mode or disabled by user.

◆ WDT is reset by

- ✓ User S/W set WDTR bit in IFF[12]. WDTR bit is automatically cleared by H/W after WDT is reset.
- ✓ Internal reset caused by any source is activated.
- ✓ Entering SLEEP mode.
- ✓ Start of FLASH programming (erase/write) by IAP.

◆ Run Control of WDT

- ✓ WDT may be disabled if WDTE flag in IFF[13] is cleared.
- ✓ When disabled WDT holds the state before.
- ✓ User can modify WDTE if and only if MAP1 flag in IFF[11] is set and MAPO flag in IFF[10] is cleared.
- ✓ WDTE is set by internal reset and also set by H/W when user sets SLEEP flag in IFF[14] or writes IAPCON SFR.

◆ Program Sequence to disable WDT

```
MOV L, #11
SETB @L ; Enable MAP1
MOV L, #13
CLR @L ; Disable WDT
MOV L, #11
CLR @L ; Disable MAP1
```

[Example of WDT Period]

XT/RG	DIV2	DIV1	DIV0	F _{osc} (MHz)	F _{sys}	WDT Period (ms)
1	0	1	1	3.64	F _{osc} /8	288
0	0	0	0	7.28	F _{osc}	18
0	1	1	0	7.28	F _{osc} /64	1152

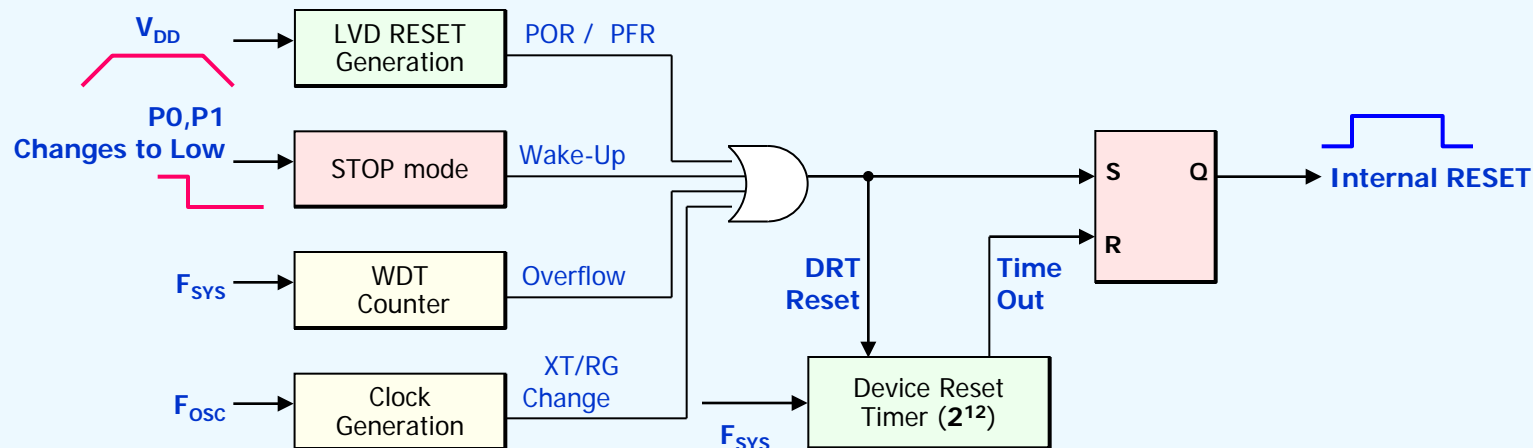
6.10. Reset Circuit

◆ Reset Sources

- ✓ Power-on Reset (POR) when Power-Up.
- ✓ Power-fail Reset
- ✓ STOP mode Wake-up by changes in input port P0 or P1.
- ✓ WDT Overflow for abnormal condition or SLEEP mode.
- ✓ Clock source change (State change of CKCFG[3]).

◆ Device Reset Timer

- ✓ Once set, internal reset remains high until the DRT (Device Reset Timer) is expired.
- ✓ The reset time depends on the configuration of system clock in CKCFG SFR.
- ✓ For an instance, the period for 2^{12} is 9 ms when F_{SYS} is 455 KHz.
- ✓ Note that CKCFG is not affected by internal reset.
- ✓ For power-on reset, the reset time is about 10 ms.



6.11. Power Management : 3 Modes

- ◆ Active Mode
 - ✓ CPU and peripheral are running.
- ◆ Sleep Mode
 - ✓ Only WDT is running.
 - ✓ I/O ports hold the state before sleep mode.
 - ✓ Wake-up by WDT overflow.
 - ✓ The longest period of WDT overflow is 1.1 second when the internal RING clock is used.
 - ✓ Device is reset.
- ◆ Stop Mode
 - ✓ All of the device function including external clock oscillator stops running.
 - ✓ I/O ports hold the state before stop mode.
 - ✓ Wake-up by input pin (P0, P1) changes.
 - ✓ Device is reset.

6.12. In Application Programming (IAP)

◆ In Application Programming

- ✓ User S/W can read or modify specific regions of FLASH with IAP function during operation.
- ✓ The EEP0/1 regions may be used as program memory or data memory.
- ✓ CPU is halt during IAP and continues execution after IAP from the next instruction which set IAPCON.
- ✓ It takes 6 system clocks to read a byte with IAP.
- ✓ It takes about 2 ms to write(erase) a byte with IAP.
- ✓ When user attempts to write IAPCON, WDTE bit in IFF[13] is also set.
- ✓ If IAP operation is erase or write, WDT is reset before the programming is started.

◆ IAP Related SFR

- ✓ DPH / DPL : Least significant 6-bit address for IAP.
- ✓ GDH / GDL : 8-bit data buffer for read or write by IAP.
- ✓ IAPCON : IAP control SFR. Automatically cleared to zero after IAP is done.

◆ IAP Enable Condition

- ✓ IAP can not erase or write INFO region.
- ✓ IAPCON can be written if and only if
 - MAPO bit in IFF[10] is cleared,
 - MAP1 bit in IFF[11] is set,
 - and corresponding bit in CFGWD[2:1] is set.
- ✓ When IAP is blocked by above condition, "MOV IAPCON, A" instruction is like "NOP" instruction.

✓ IAPCON (09h) : IAP Control Register

RGS1	RGS0	OPS1	OPS0
R/W(0)	R/W(0)	R/W(0)	R/W(0)

- RGS[1:0] : Select IAP region
- OPS[1:0] : Select IAP function

RGS1	RGS0	IAP Region
0	0	EEP0 (0x1C0 ~ 0x1FF)
0	1	EEP1 (0x3C0 ~ 0x3FF)
1	0	INFO (0x0 ~ 0x7)
1	1	Reserved

OPS1	OPS0	IAP Function
0	0	No operation
0	1	Byte Read
1	0	Byte Erase
1	1	Byte Write

6.12. In Application Programming (IAP)

Preliminary

◆ Electrical Characteristic of IAP

- ✓ Note that the program time depends on the configuration of system clock frequency.
- ✓ If the system clock frequency is out of IAP range, user need to change F_{SYS} before and after IAP by configuring CKCFG SFR.

Parameter	Symbol	MIN	TYP	MAX	Unit
Power Supply Voltage	V_{DD}	2.7	-	5.5	V
System Clock Frequency	F_{SYS}	5	8	11	MHz
Write /Erase Time	T_p	1.5	2.0	3.3	ms

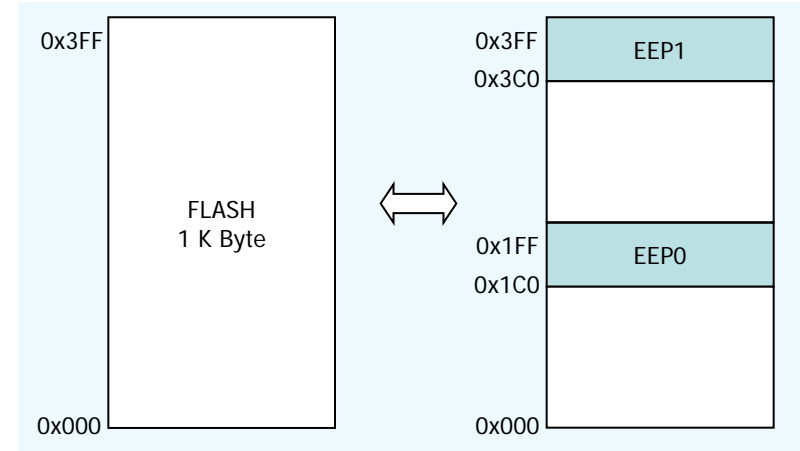
◆ Information Region

ADDRESS	0	1	2	3	4	5	6	7
Mnemonic	CFGWD							

- ✓ The first byte contains CFGWD
- ✓ May be used to store user ID, or checksum, etc.
- ✓ Only the full chip erase function of ISP can erase this region.

◆ FLASH Regions

- ✓ EEPROM area is a part of program memory.



◆ CFGWD : Configuration Word

- ✓ CFGWD[0] (ISP_LOCK) : Disable read, write, or erase by ISP except the full chip erase.
- ✓ CFGWD[1] (IAP_RE) : Enable read by IAP.
- ✓ CFGWD[2] (IAP_PE) : Enable write or erase by IAP.

7. Absolute Maximum Ratings

◆ Absolute Maximum Ratings

Symbol	Parameter	Rating	Unit
V_{DD}	DC supply voltage	-0.5 to 6.5	V
V_{IN}	DC input voltage	-0.5 to $V_{DD}+0.5$	V
V_{OUT}	DC output voltage	-0.5 to $V_{DD}+0.5$	V
I_{OH}	DC output high current	One I/O pin active : -25	mA
		All I/O pin active : -100	mA
I_{OL}	DC output low current	One I/O pin active : 30	mA
		All I/O pin active : 150	mA
T_{STG}	Storage temperature	-55 to 125	°C

◆ Recommended Operating Conditions

Symbol	Parameter	Rating	Unit
V_{DD}	DC supply voltage	1.8 to 5.5	V
T_A	Industrial temperature range	-40 to 85	°C

8. DC Characteristics

* TA = -40 °C ~ +85 °C, V_{DD} = 1.8V ~ 5.5V unless otherwise specified.

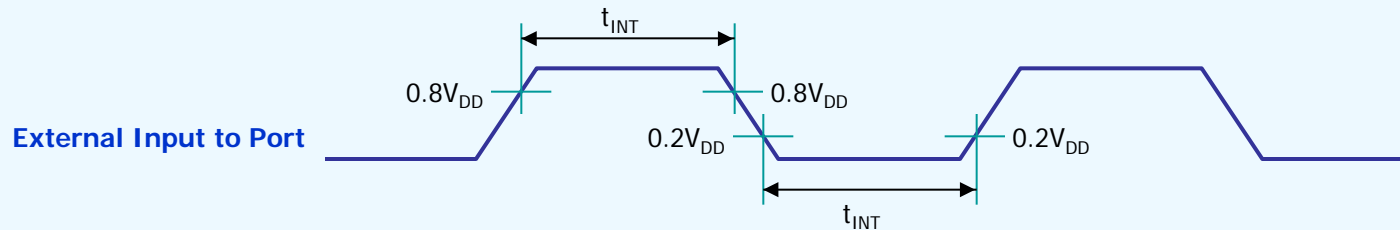
Parameter	Symbol	Pin	Conditions	Value			Unit
				Min.	Typ.	Max.	
Input Low Voltage	V _{IL1}	P0, P1, P2, P3	V _{DD} = 1.8V~5.5V	-0.5	-	0.2V _{DD} -0.1	V
Input high Voltage	V _{IH1}	P0, P1, P2, P3	V _{DD} = 1.8V~5.5V	0.2V _{DD} +1.0	-	V _{DD} +0.5	V
Input High Leakage Current	I _{IH}	All pins except XI, XO	V _{IN} = V _{DD}	-1	-	+1	μA
Output Low Voltage	V _{OL}	P0, P1, P2, P3	I _{OL} = 20mA @V _{DD} =5V (I _{OL} = 8mA @V _{DD} =3V, I _{OL} = 3mA @V _{DD} =2.2V)	-	-	0.3V _{DD}	V
Output Low Voltage	V _{OL2}	REM	I _{OL2} = 280mA @V _{DD} =3V (I _{OL2} = 180mA @V _{DD} =1.8V)	-	-	0.4	V
Output High Voltage	V _{OH}	P2 (Configured as push-pull output)	I _{OH} = -0.5mA @V _{DD} =5V (I _{OH} = -0.3mA @V _{DD} =3V)	0.7V _{DD}	-	-	V
Output High Voltage	V _{OHP}	Pull-up current	I _{OHP} = -100uA @V _{DD} =5V (I _{OHP} = -50uA @V _{DD} =3V)	0.7V _{DD}	-	-	V
Pin Capacitance	C _{IO}	All	V _{DD} = 5V	-	10	-	pF

9. AC Characteristics

Preliminary

* TA = -40 °C ~ +85 °C unless otherwise specified. TBD = To Be Determined.

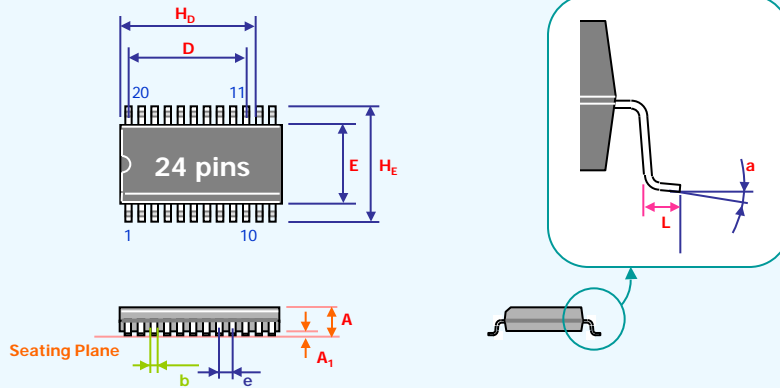
Parameter	Symbol	Pin	Conditions	Value			Unit
				Min.	Typ.	Max.	
Oscillator Frequency (Internal Clock)	F _{osc}		2.7 V ≤ V _{DD} ≤ 5.5 V	-		10	MHz
			1.8 V ≤ V _{DD} < 2.7 V	-		5	
Oscillator Frequency (External Clock)	F _{osc}	XI, XO	2.7 V ≤ V _{DD} ≤ 5.5 V	-	-	10	MHz
			1.8 V ≤ V _{DD} < 2.7 V	-	-	5	
System Frequency	F _{sys}		1.8 V ≤ V _{DD} ≤ 5.5 V	1/64	-	1	F _{osc}
External Input Width	t _{INT}	P0, P1, P2, P3, P4	1.8 V ≤ V _{DD} ≤ 5.5 V	12	-	-	F _{sys}



10. Package Dimensions

Preliminary

[24-SOIC]

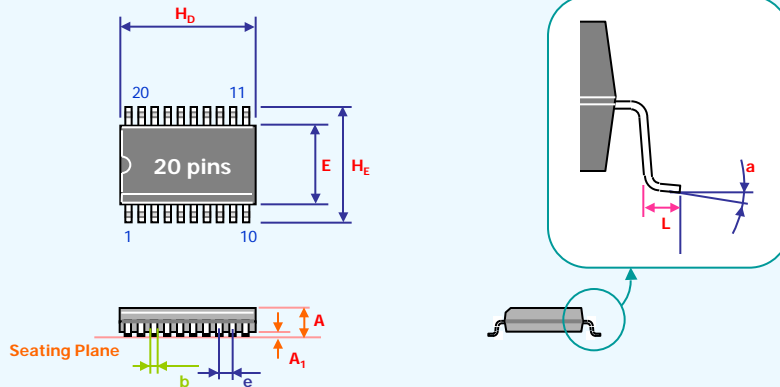


Symbol	Dimension in Inches			Dimension in mm		
	Min.	Nom.	Max.	Min.	Nom.	Max.
A	0.094	0.098	0.102	2.40	2.50	2.60
A ₁	0.004	0.008	0.012	0.10	0.20	0.30
b	0.014	0.017	0.019	0.36	0.42	0.49
D	-	0.550	-	-	13.97	-
E	0.291	0.295	0.299	7.40	7.50	7.60
H _b	0.598	0.606	0.614	15.20	15.40	15.60
H _e	0.398	0.406	0.413	10.10	10.30	10.50
L	0.004	0.010	0.016	0.10	0.25	0.40
a	0°	-	8°	0°	-	8°
e	0.050 BSC			1.27 BSC		

Notes:

1. Dimension D & E include mold mismatch and are determined at the mold parting line.
2. General appearance spec. should be based on final visual inspection spec.

[20-SOIC (JEDEC)]



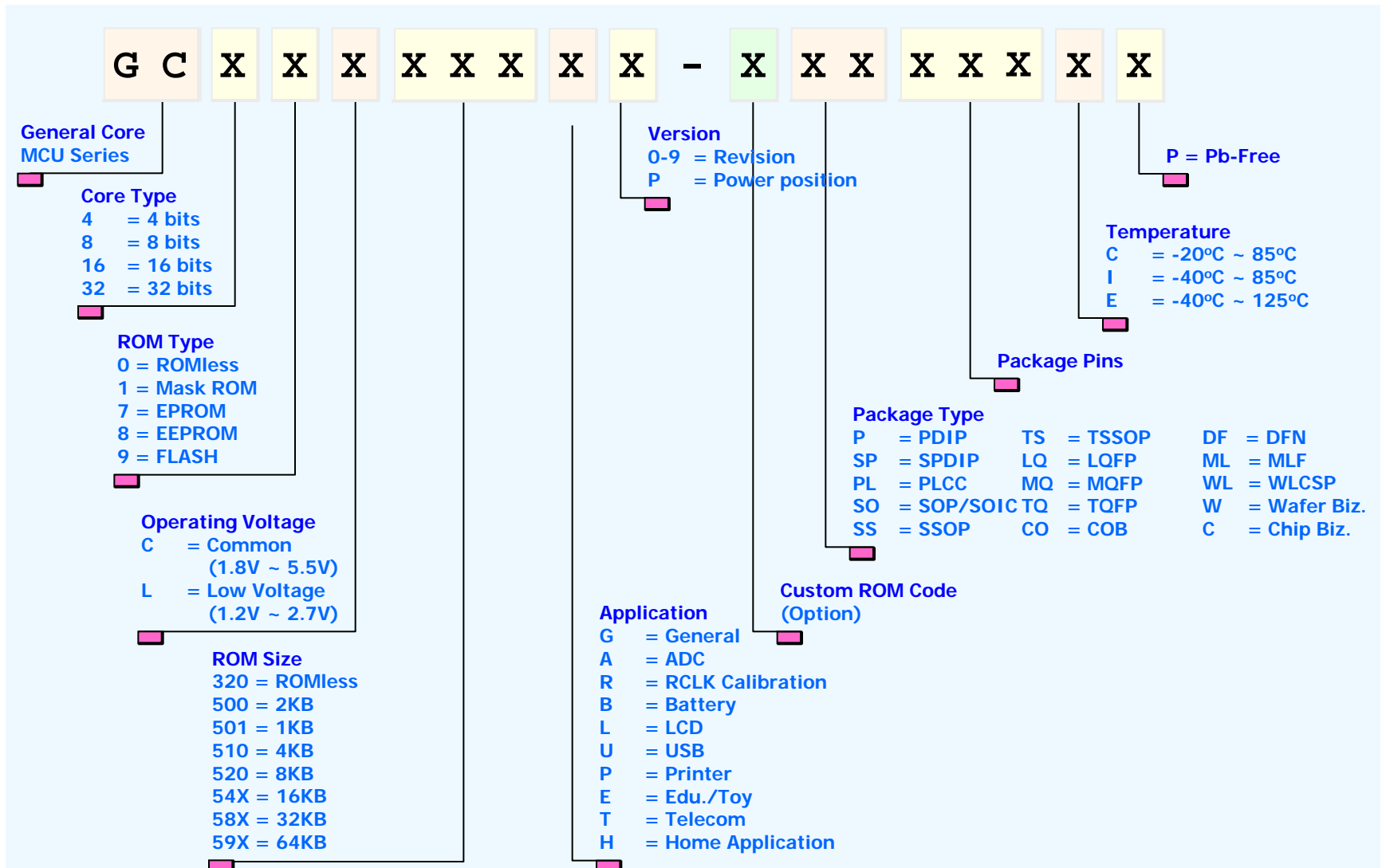
Symbol	Dimension in Inches			Dimension in mm		
	Min.	Nom.	Max.	Min.	Nom.	Max.
A	-	-	0.106	-	-	2.7
A ₁	0.004	-	-	0.1	-	-
b	0.013	0.016	0.020	0.324	0.4	0.51
E	0.264	0.295	0.324	6.71	7.5	8.23
H _b	0.495	0.504	0.512	12.57	12.8	13
H _e	0.394	0.406	0.419	10.0	10.3	10.643
L	0.016	-	0.052	0.406	-	1.32
a	0°	-	8°	0°	-	8°
e	0.050 BSC			1.27 BSC		

Notes:

1. Dimension D & E include mold mismatch and are determined at the mold parting line.
2. General appearance spec. should be based on final visual inspection spec.

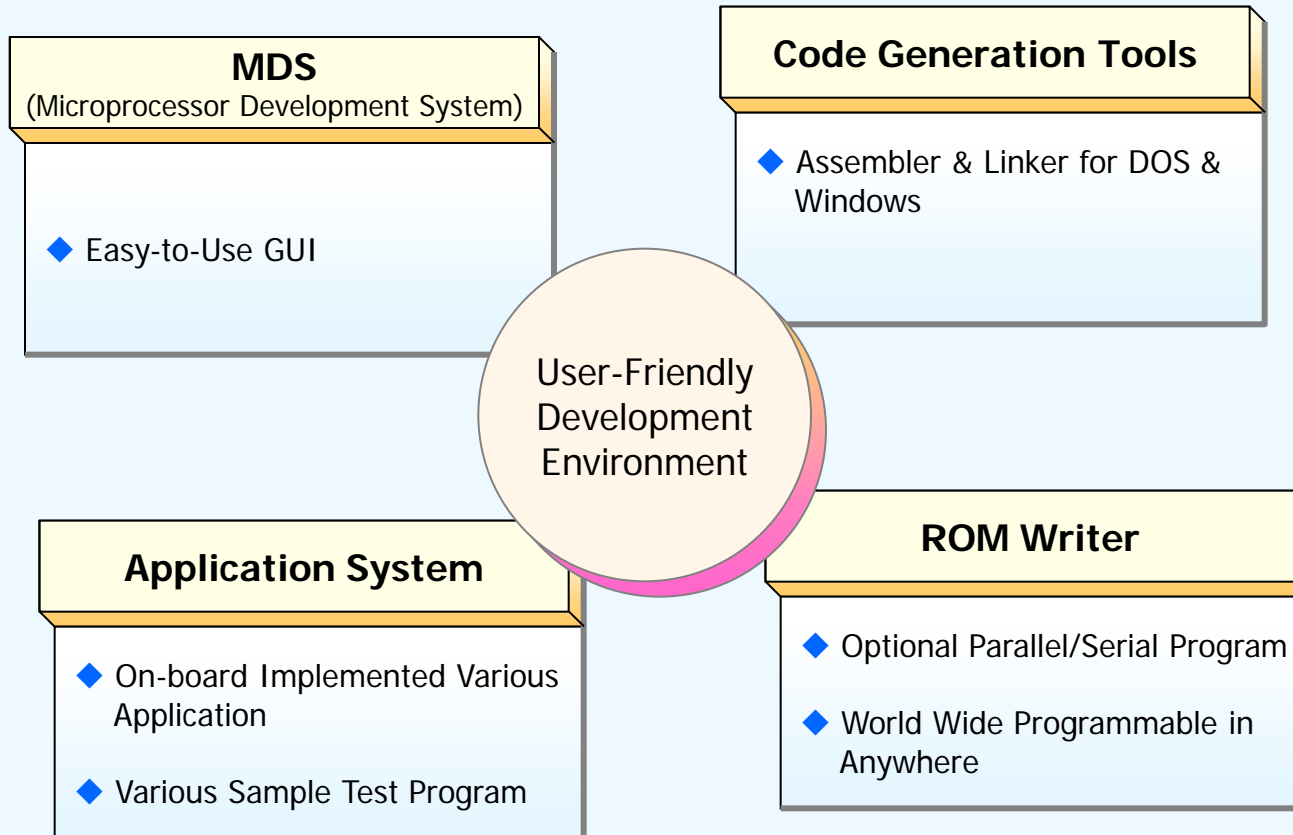
11. Product Numbering System

Preliminary



12. Supporting tools

Preliminary



Appendix A : Instruction Set (1/19)

◆ Abbreviations and Symbols

Symbol	Description	Symbol	Description
PC	The program counter.	(PC)	The contents of PC.
A	The accumulator register (ACC).	(A)	The contents of ACC.
C	The carry flag.	(C)	The contents of C.
SP	The stack pointer register. Concatenation of SPH and SPL.	M[SP]	The contents of RAM addressed by SP.
(DP)	The contents of DPTR.	(SP)	The contents of SP.
DP	The data pointer register (DPTR). Concatenation of DPH and DPL.	M[DP]	The contents of RAM addressed by DPTR.
H	The high nibble of the data pointer (DPH).	(H)	The contents of DPH.
L	The low nibble of the data pointer (DPL).	(L)	The contents of DPL.
F[L]	The contents of indirect function flag (IFF) addressed by DPL.	rel	8-bit signed displacement value for relative branch ($-128 \leq \text{rel} \leq 127$).
#data	4-bit data operand	addr	12-bit absolute branch address.
dir	4-bit direct address of SFRs ($0 \leq \text{dir} \leq 15$)	R[dir]	The contents of SFR or read value of ports.
bit	2-bit pointer of the bit in data memory addressed by DPTR ($0 \leq \text{bit} \leq 3$).	M[DP].bit	The value of memory bit which is addressed by DPTR and bit.
@	Prefix for indirect address	Pm.n	Value of bit n of I/O port m.
\leq	Less than or equal to	.	Value of PC for current instruction.
\leftarrow	Transfer	\leftrightarrow	Exchange
=	Equal to	\neq	Not equal to
$>$	Greater than	$<$	Less than
+	Addition	-	Subtraction
&	Bitwise logical AND		Bitwise logical OR
^	Bitwise logical Exclusive-OR	\sim	Bitwise logical complement
{b,b}	Concatenation of bits		

Appendix A : Instruction Set (2/19)

Preliminary

◆ OP CODE Map

H \ L	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	NOP	SETB C	PUSH A	POP A	INC DPTR	DEC DPTR	INC @DP	DEC @DP	ADD A, @DP	ADDC A, @DP	CPL A	SUB A, @DP	ANL A, @DP	ORL A, @DP	XRL A, @DP	RRC A
1	CLR C	INC A	ADD A, #data													DEC A
2	MOV L, #data															
3	MOV H, #data															
4	MOVI @DP, #data															
5	CLR A	MOV A, #data														
6	MOV dir, A															
7	MOV A, dir															
8	MOV A, @DP	XCH A, @DP	MOV L, @DP	MOV @DP, A	MOVI @DP, A	MOV D @DP, A	CLR @L	SETB @L	CLR bit				SETB bit			
9	RET	DJNZ A, rel	CJNE A, @DP, rel	CJLE A, @DP, rel			JNC rel	JC rel	JNB bit, rel			JB bit, rel				
A	CJNE L, #data, rel															
B	CJNE @DP, #data, rel															
C	CJNE A, #data, rel															
D	CJNE A, dir, rel															
E	JMP addr															
F	CALL addr															

ADD A, #data

Binary Code	<table border="1"><tr><td>0001</td><td>dddd</td></tr></table>	0001	dddd
0001	dddd		
Description	Adds the 4-bit data to the Accumulator. The result is stored in Accumulator. When adding unsigned integers, the carry flag indicates an overflow.		
Operation	$(A) \leftarrow (A) + \#data$		
Carry Flag	Set if a carry occurred, cleared otherwise.		
Bytes	1		
Cycles	1		
Example	CLR A ; Clear ACC ADD A, #2 ; Add 2 to ACC. ACC contains 2.		

ADD A, @DP

Binary Code	<table border="1"><tr><td>0000</td><td>1000</td></tr></table>	0000	1000
0000	1000		
Description	Adds the contents of indirect data memory to the Accumulator. The result is stored in Accumulator. When adding unsigned integers, the carry flag indicates an overflow.		
Operation	$(A) \leftarrow (A) + M[DP]$		
Carry Flag	Set if a carry occurred, cleared otherwise.		
Bytes	1		
Cycles	1		
Example	; Assumes M[DP] contains 2 MOV A, #8 ; Set ACC as 8. ADD A, @DP ; The result, 10 is stored in ACC.		

ADDC A, @DP

Binary Code	<table border="1"><tr><td>0000</td><td>1001</td></tr></table>	0000	1001
0000	1001		
Description	Simultaneously adds the contents of indirect data memory, the carry flag and the Accumulator. The result is stored in Accumulator. When adding unsigned integers, the carry flag indicates an overflow.		
Operation	$(A) \leftarrow (A) + M[DP] + (C)$		
Carry Flag	Set if a carry occurred, cleared otherwise.		
Bytes	1		
Cycles	1		
Example	; Assumes M[DP] contains 2 and C is 1. MOV A, #8 ; Set ACC as 8. ADDC A, @DP ; The result, 11 is stored in ACC.		

ANL A, @DP

Binary Code	0000	1100
Description	ANL performs the bitwise logical-AND operation between the indirect data memory and ACC. The result is stored in Accumulator.	
Operation	$(A) \leftarrow (A) \& M[DP]$	
Carry Flag	Not affected.	
Bytes	1	
Cycles	1	
Example	; Assumes M[DP] contains 2 MOV A, #0xA ; Set ACC as 10. ANL A, @DP ; The result, 2 is stored in ACC.	

CALL addr

Binary Code	1111	aaaa	aaaa	aaaa
Description	Unconditionally calls a subroutine located at the indicated 12-bit address. The instruction increments the PC twice to obtain the address of the following instruction, then push the result onto the stack (low-order nibble first). The stack pointer is incremented three times. The destination address is obtained by concatenating four low-order bits of the opcode byte and the second byte of the instruction.			
Operation	$(PC) \leftarrow (PC) + 2$ $(SP) \leftarrow (SP) + 1$ $M[SP] \leftarrow (PC_{3-0})$ $(SP) \leftarrow (SP) + 1$ $M[SP] \leftarrow (PC_{7-4})$ $(SP) \leftarrow (SP) + 1$ $M[SP] \leftarrow (PC_{11-8})$ $(PC) \leftarrow \text{addr}$			
Carry Flag	Not affected.			
Bytes	2			
Cycles	2			
Example	CALL SUBR ; Call subroutine located ; at the label SUBR.			

CJLE A, @DP, rel

Binary Code	1001	0011	rrrr	rrrr
Description	<p>Compares the contents of ACC and the indirect memory, and branches if the value in ACC is less than or equal to that in memory.</p> <p>The branch destination is computed by adding the signed relative-displacement in the second byte of the instruction to the PC, after incrementing the PC to the start of the next instruction. The contents of both operands are not affected by comparison.</p> <p>The carry flag is set if the contents are equal.</p>			
Operation	$(PC) \leftarrow (PC) + 2$ IF $(A) \leq M[DP]$ THEN $(PC) \leftarrow (PC) + rel$			
Carry Flag	IF $(A) = M[DP]$ THEN $(C) \leftarrow 1$ ELSE $(C) \leftarrow 0$.			
Bytes	2			
Cycles	2			
Example	; Assumes M[DP] contains 11, ACC 5. CJLE A, @DP, CMP_LE; Branches to CMP_LE ; IF $(A) > M[DP]$ CMP_LE: JC CMP_EQ ; ; IF $(A) < M[DP]$ CMP_EQ: ; IF $(A) = M[DP]$			

CJNE @DP, #data, rel

Binary Code	1011	dddd	rrrr	rrrr
Description	<p>Compares the contents of the indirect memory and data in four low-order bits of opcode, and branches if their values are not equal.</p> <p>The branch destination is computed by adding the signed relative-displacement in the second byte of the instruction to the PC, after incrementing the PC to the start of the next instruction. The contents of indirect memory is not affected.</p> <p>The carry flag is set if the unsigned integer value of M[DP] is less than the unsigned integer value of the data; otherwise, the carry is cleared.</p>			
Operation	$(PC) \leftarrow (PC) + 2$ IF $M[DP] \neq \#data$ THEN $(PC) \leftarrow (PC) + rel$			
Carry Flag	IF $M[DP] < \#data$ THEN $(C) \leftarrow 1$ ELSE $(C) \leftarrow 0$.			
Bytes	2			
Cycles	2			
Example	; Assumes M[DP] contains 2. CJNE @DP, #8, CMP_NE; Branches to CMP_NE ; IF $M[DP] = 8$ CMP_NE: JC CMP_LT ; Branches to CMP_LT ; IF $M[DP] > 8$ CMP_LT: ; IF $M[DP] < 8$			

CJNE A, dir, rel

Binary Code	<table border="1" style="display: inline-table; border-collapse: collapse;"><tr><td style="padding: 2px 10px;">1101</td><td style="padding: 2px 10px;">dddd</td><td style="padding: 2px 10px;">rrrr</td><td style="padding: 2px 10px;">rrrr</td></tr></table>	1101	dddd	rrrr	rrrr
1101	dddd	rrrr	rrrr		
Description	<p>Compares the contents of ACC and that of SFR addressed by four low-order bits of opcode, and branches if their values are not equal.</p> <p>The branch destination is computed by adding the signed relative-displacement in the second byte of the instruction to the PC, after incrementing the PC to the start of the next instruction. The contents of both operands are not affected by comparison.</p> <p>The carry flag is set if the unsigned integer value of ACC is less than the unsigned integer value of the SFR; otherwise, the carry is cleared.</p>				
Operation	$(PC) \leftarrow (PC) + 2$ IF $(A) \neq R[dir]$ THEN $(PC) \leftarrow (PC) + rel$				
Carry Flag	IF $(A) < R[dir]$ THEN $(C) \leftarrow 1$ ELSE $(C) \leftarrow 0$.				
Bytes	2				
Cycles	2				
Example	; Wait until P0 (Port 0) is 0xE. MOV A, #0xE CJNE A, P0, . ; Self looping with "."				

CJNE L, #data, rel

Binary Code	<table border="1" style="display: inline-table; border-collapse: collapse;"><tr><td style="padding: 2px 10px;">1010</td><td style="padding: 2px 10px;">dddd</td><td style="padding: 2px 10px;">rrrr</td><td style="padding: 2px 10px;">rrrr</td></tr></table>	1010	dddd	rrrr	rrrr
1010	dddd	rrrr	rrrr		
Description	<p>Compares the contents of DPL and data in four low-order bits of opcode, and branches if their values are not equal.</p> <p>The branch destination is computed by adding the signed relative-displacement in the second byte of the instruction to the PC, after incrementing the PC to the start of the next instruction. The contents of DPL is not affected.</p> <p>The carry flag is set if the unsigned integer value of DPL is less than the unsigned integer value of the data; otherwise, the carry is cleared.</p>				
Operation	$(PC) \leftarrow (PC) + 2$ IF $(L) \neq \#data$ THEN $(PC) \leftarrow (PC) + rel$				
Carry Flag	IF $(L) < \#data$ THEN $(C) \leftarrow 1$ ELSE $(C) \leftarrow 0$.				
Bytes	2				
Cycles	2				
Example	; Looping with DPL MOV L, #9 ; $(L) \leftarrow 9$ LOOP_L: ; Operations in loop ; Operations in loop DEC DPTR ; $(DP) \leftarrow (DP) - 1$ CJNE L, #0, LOOP_L ; Repeat until (L) is 0.				

CLR @L

Binary Code	<table border="1"><tr><td>1000</td><td>0110</td></tr></table>	1000	0110
1000	0110		
Description	Clears the indirect function flag addressed by DPL.		
Operation	$F[L] \leftarrow 0$		
Carry Flag	Not affected.		
Bytes	1		
Cycles	1		
Example	<pre> ; Assumes P2 contains 0xF. MOV L, #1 ; (L) ← 1 CLR @L ; P2.1 ← 0 MOV A, #0xD ; (A) ← 13 CJNE A, P2, ERROR ; Check if P2.1 is 0. </pre>		

CLR A

Binary Code	<table border="1"><tr><td>0101</td><td>0000</td></tr></table>	0101	0000
0101	0000		
Description	Clears the accumulator. This is an abbreviation of MOV A, #0.		
Operation	$(A) \leftarrow 0$		
Carry Flag	Not affected.		
Bytes	1		
Cycles	1		
Example	CLR A		

CLR C

Binary Code	<table border="1"><tr><td>0001</td><td>0000</td></tr></table>	0001	0000
0001	0000		
Description	Clears the carry flag. This is the same as "ADD A, #0".		
Operation	$(A) \leftarrow (A) + 0$		
Carry Flag	$(C) \leftarrow 0$		
Bytes	1		
Cycles	1		
Example	CLR C		

CLR bit

Binary Code	<table border="1"><tr><td>1000</td><td>10bb</td></tr></table>	1000	10bb
1000	10bb		
Description	Clears a bit in data memory addressed by DPTR. The bit position of the nibble is obtained by the least significant two bits of opcode.		
Operation	$M[DP].bit \leftarrow 0$		
Carry Flag	Not affected.		
Bytes	1		
Cycles	1		
Example	<pre> ; Assumes M[DP] contains 7. CLR 2 ; M[DP].2 ← 0 CJNE @DP, #3, ERROR ; Check result </pre>		

CPL A

Binary Code	<table border="1"><tr><td>0000</td><td>1010</td></tr></table>	0000	1010
0000	1010		
Description	Complements the contents of ACC.		
Operation	$(A) \leftarrow \sim(A)$		
Carry Flag	Not affected.		
Bytes	1		
Cycles	1		
Example	MOV A, P0 ; (A) ← P0 CPL A ; ACC contains 1's ; complement of P0		

DEC @DP

Binary Code	<table border="1"><tr><td>0000</td><td>0111</td></tr></table>	0000	0111
0000	0111		
Description	Decrements the value of data memory addressed indirectly by DPTR.		
Operation	$M[DP] \leftarrow M[DP] - 1$		
Carry Flag	Not affected.		
Bytes	1		
Cycles	1		
Example	DEC @DP		

DEC A

Binary Code	<table border="1"><tr><td>0001</td><td>1111</td></tr></table>	0001	1111
0001	1111		
Description	Decrements the contents of ACC. This is the same as "ADD A, #15". Carry is cleared when the borrow occurs; otherwise, carry is set.		
Operation	$(A) \leftarrow (A) + 15$		
Carry Flag	IF (A) = 0 THEN C ← 0 ELSE C ← 1.		
Bytes	1		
Cycles	1		
Example	DEC A		

DEC DPTR

Binary Code	<table border="1"><tr><td>0000</td><td>0101</td></tr></table>	0000	0101
0000	0101		
Description	Decrements the data pointer.		
Operation	$(DP) \leftarrow (DP) - 1$		
Carry Flag	Not affected.		
Bytes	1		
Cycles	1		
Example	; Assumes DPTR contains 0. DEC DPTR ; By underflow, all bits ; of DPH and DPL are set. DEC DP ; This is also valid.		

DJNZ A, rel

Binary Code	<table border="1"><tr><td>1001</td><td>0001</td><td>rrrr</td><td>rrrr</td></tr></table>	1001	0001	rrrr	rrrr
1001	0001	rrrr	rrrr		
Description	<p>Decrements the contents of ACC, and branches if the result is not zero.</p> <p>The branch destination is computed by adding the signed relative-displacement in the second byte of the instruction to the PC, after incrementing the PC to the start of the next instruction.</p> <p>Carry is cleared when the borrow occurs; otherwise, carry is set.</p>				
Operation	$(PC) \leftarrow (PC) + 2$ $(A) \leftarrow (A) - 1$ IF $(A) \neq 0$ THEN $(PC) \leftarrow (PC) + rel$				
Carry Flag	IF $(A) = 0$ THEN $(C) \leftarrow 0$ ELSE $(C) \leftarrow 1$.				
Bytes	2				
Cycles	2				
Example	MOV A, @DP DJNZ A, ACC_NZ ACC_NZ: JNC ACC_ZERO				

INC @DP

Binary Code	<table border="1"><tr><td>0000</td><td>0110</td></tr></table>	0000	0110
0000	0110		
Description	Increases the value of data memory addressed indirectly by DPTR.		
Operation	$M[DP] \leftarrow M[DP] + 1$		
Carry Flag	Not affected.		
Bytes	1		
Cycles	1		
Example	INC @DP		

INC A

Binary Code	<table border="1"><tr><td>0001</td><td>0001</td></tr></table>	0001	0001
0001	0001		
Description	<p>Increases the contents of ACC.</p> <p>This is the same as "ADD A, #1".</p> <p>Carry is set when the overflow occurs; otherwise, carry is cleared.</p>		
Operation	$(A) \leftarrow (A) + 1$		
Carry Flag	IF $(A) = 15$ THEN $C \leftarrow 1$ ELSE $C \leftarrow 0$.		
Bytes	1		
Cycles	1		
Example	INC A		

INC DPTR

Binary Code	<table border="1"><tr><td>0000</td><td>0100</td></tr></table>	0000	0100
0000	0100		
Description	Increments the data pointer.		
Operation	$(DP) \leftarrow (DP) + 1$		
Carry Flag	Not affected.		
Bytes	1		
Cycles	1		
Example	<pre> ; Assumes all bits of DPTR is 1. INC DPTR ; By roll over, all bits ; of DPH and DPL are cleared. INC DP ; This is also valid. </pre>		

JB bit, rel

Binary Code	<table border="1"><tr><td>1001</td><td>11bb</td><td>rrrr</td><td>rrrr</td></tr></table>	1001	11bb	rrrr	rrrr
1001	11bb	rrrr	rrrr		
Description	<p>Branches if the bit in data memory is 1. The address is given by DPTR and bit position is given by two least significant bits of opcode .</p> <p>The branch destination is computed by adding the signed relative-displacement in the second byte of the instruction to the PC, after incrementing the PC to the start of the next instruction. The contents of memory is not affected.</p>				
Operation	$(PC) \leftarrow (PC) + 2$ IF M[DP].bit = 1 THEN $(PC) \leftarrow (PC) + rel$				
Carry Flag	Not affected.				
Bytes	2				
Cycles	2				
Example	<pre> JB 0, L_BIT_SET ; IF M[DP].0 = 0 L_BIT_SET: ; IF M[DP].0 = 1 </pre>				

JC rel

Binary Code	1001	0111	rrrr	rrrr
Description	Branches if the carry flag is 1. The branch destination is computed by adding the signed relative-displacement in the second byte of the instruction to the PC, after incrementing the PC to the start of the next instruction.			
Operation	$(PC) \leftarrow (PC) + 2$ IF (C) = 1 THEN $(PC) \leftarrow (PC) + rel$			
Carry Flag	Not affected.			
Bytes	2			
Cycles	2			
Example	JC L_C_SET ; IF (C) = 0 L_C_SET: ; IF (C) = 1			

JMP addr

Binary Code	1110	aaaa	aaaa	aaaa
Description	Transfers program execution to the indicated 12-bit address. The destination address is obtained by concatenating the four low-order bits of the opcode byte and the second byte of the instruction.			
Operation	$(PC) \leftarrow addr$			
Carry Flag	Not affected.			
Bytes	2			
Cycles	2			
Example	JMP LABEL ; Jumps to LABEL. JMP . ; Infinite loop			

JNB bit, rel

Binary Code	<table border="1"><tr><td>1001</td><td>10bb</td><td>rrrr</td><td>rrrr</td></tr></table>	1001	10bb	rrrr	rrrr
1001	10bb	rrrr	rrrr		
Description	<p>Branches if the bit in data memory is 0.</p> <p>The address of memory is given by DPTR and bit position is given by two least significant bits of opcode .</p> <p>The branch destination is computed by adding the signed relative-displacement in the second byte of the instruction to the PC, after incrementing the PC to the start of the next instruction. The contents of memory is not affected.</p>				
Operation	$(PC) \leftarrow (PC) + 2$ IF $M[DP].bit = 0$ THEN $(PC) \leftarrow (PC) + rel$				
Carry Flag	Not affected.				
Bytes	2				
Cycles	2				
Example	JNB 3, L_BIT_ZERO ; IF $M[DP].3 = 1$ L_BIT_ZERO: ; IF $M[DP].3 = 0$				

JNC rel

Binary Code	<table border="1"><tr><td>1001</td><td>0110</td><td>rrrr</td><td>rrrr</td></tr></table>	1001	0110	rrrr	rrrr
1001	0110	rrrr	rrrr		
Description	<p>Branches if the carry flag is 0.</p> <p>The branch destination is computed by adding the signed relative-displacement in the second byte of the instruction to the PC, after incrementing the PC to the start of the next instruction.</p>				
Operation	$(PC) \leftarrow (PC) + 2$ IF $(C) = 0$ THEN $(PC) \leftarrow (PC) + rel$				
Carry Flag	Not affected.				
Bytes	2				
Cycles	2				
Example	JNC L_C_ZERO ; IF $(C) = 1$ L_C_ZERO: ; IF $(C) = 0$				

MOV @DP, A

Binary Code	<table border="1"><tr><td>1000</td><td>0011</td></tr></table>	1000	0011
1000	0011		
Description	The contents of ACC is copied to data memory whose address is given by DPTR.		
Operation	$M[DP] \leftarrow (A)$		
Carry Flag	Not affected.		
Bytes	1		
Cycles	1		
Example	<pre>MOV H, #2 ; (H) ← 2 MOV L, #14 ; (L) ← 14 MOV @DP, A</pre>		

MOV A, @DP

Binary Code	<table border="1"><tr><td>1000</td><td>0000</td></tr></table>	1000	0000
1000	0000		
Description	Copies the contents of data memory to ACC. The address of memory is given by DPTR.		
Operation	$(A) \leftarrow M[DP]$		
Carry Flag	Not affected.		
Bytes	1		
Cycles	1		
Example	<pre>MOV H, #1 ; (H) ← 1 MOV L, #0 ; (L) ← 0 MOV A, @DP</pre>		

MOV A, #data

Binary Code	<table border="1"><tr><td>0101</td><td>dddd</td></tr></table>	0101	dddd
0101	dddd		
Description	Sets ACC with the data given in four low-order bits of opcode.		
Operation	$(A) \leftarrow \#data$		
Carry Flag	Not affected.		
Bytes	1		
Cycles	1		
Example	<pre>MOV A, #-1 ; (A) ← 15 MOV A, #0xC ; (A) ← 12</pre>		

MOV A, dir

Binary Code	<table border="1"><tr><td>0111</td><td>dddd</td></tr></table>	0111	dddd
0111	dddd		
Description	The contents of SFR is copied to ACC. The address of SFR is given by four low-order bits of opcode.		
Operation	$(A) \leftarrow R[dir]$		
Carry Flag	Not affected.		
Bytes	1		
Cycles	1		
Example	<pre>MOV A, P0 ; Read Port-0 into ACC. MOV A, L ; Move DPL to ACC. MOV A, SPH ; Move SPH to ACC.</pre>		

MOV H, #data

Binary Code	<table border="1"><tr><td>0011</td><td>dddd</td></tr></table>	0011	dddd
0011	dddd		
Description	Sets DPH with the data given in four low-order bits of opcode.		
Operation	$(H) \leftarrow \#data$		
Carry Flag	Not affected.		
Bytes	1		
Cycles	1		
Example	MOV H, #1 ; $(H) \leftarrow 1$		

MOV L, @DP

Binary Code	<table border="1"><tr><td>1000</td><td>0010</td></tr></table>	1000	0010
1000	0010		
Description	Copies the contents of data memory to DPL. The address of memory is given by DPTR.		
Operation	$(L) \leftarrow M[DP]$		
Carry Flag	Not affected.		
Bytes	1		
Cycles	1		
Example	MOV H, #0 MOV L, #3 MOV L, @DP ; L is changed to M[DP]		

MOV L, #data

Binary Code	<table border="1"><tr><td>0010</td><td>dddd</td></tr></table>	0010	dddd
0010	dddd		
Description	Sets DPL with the data given in four low-order bits of opcode.		
Operation	$(L) \leftarrow \#data$		
Carry Flag	Not affected.		
Bytes	1		
Cycles	1		
Example	MOV L, #5 ; $(L) \leftarrow 5$		

MOV dir, A

Binary Code	<table border="1"><tr><td>0110</td><td>dddd</td></tr></table>	0110	dddd
0110	dddd		
Description	The contents of ACC is copied to SFR. The address of SFR is given by four low-order bits of opcode.		
Operation	$R[dir] \leftarrow (A)$		
Carry Flag	Not affected.		
Bytes	1		
Cycles	1		
Example	MOV P0, A ; Output ACC to Port-0. MOV H, A ; Move ACC to DPH. MOV DPH, A ; Move ACC to DPH. MOV SPL, A ; Move ACC to SPL.		

MOVD @DP, A

Binary Code	<table border="1"><tr><td>1000</td><td>0101</td></tr></table>	1000	0101
1000	0101		
Description	The contents of ACC is copied to data memory whose address is given by DPTR. After that the data pointer is decremented.		
Operation	$M[DP] \leftarrow (A)$ $(DP) \leftarrow (DP) - 1$		
Carry Flag	Not affected.		
Bytes	1		
Cycles	1		
Example	MOVD @DP, A		

MOVI @DP, A

Binary Code	<table border="1"><tr><td>1000</td><td>0100</td></tr></table>	1000	0100
1000	0100		
Description	The contents of ACC is copied to data memory whose address is given by DPTR. After that the data pointer is incremented.		
Operation	$M[DP] \leftarrow (A)$ $(DP) \leftarrow (DP) + 1$		
Carry Flag	Not affected.		
Bytes	1		
Cycles	1		
Example	MOVI @DP, A		

MOVI @DP, #data

Binary Code	<table border="1"><tr><td>0100</td><td>dddd</td></tr></table>	0100	dddd
0100	dddd		
Description	Set data memory whose address is given by DPTR with the data given in four low-order bits of opcode. After that the data pointer is incremented.		
Operation	$M[DP] \leftarrow \#data$ $(DP) \leftarrow (DP) + 1$		
Carry Flag	Not affected.		
Bytes	1		
Cycles	1		
Example	; Simple look-up of constant values MOV L, #0 ; Pointer to store MOV H, #1 ; look-up values CALL TABLE TABLE: MOVI @DP, #0xC MOVI @DP, #0x0 MOVI @DP, #0x0 MOVI @DP, #0x1 RET		

NOP

Binary Code	<table border="1"><tr><td>0000</td><td>0000</td></tr></table>	0000	0000
0000	0000		
Description	No operation. Just fetches the next instruction.		
Operation	$(PC) \leftarrow (PC) + 1$		
Carry Flag	Not affected.		
Bytes	1		
Cycles	1		
Example	NOP		

POP A

Binary Code	<table border="1"><tr><td>0000</td><td>0011</td></tr></table>	0000	0011
0000	0011		
Description	The contents of stack top is moved to ACC. After that the stack pointer is decremented by 1.		
Operation	$(A) \leftarrow M[SP]$ $(SP) \leftarrow (SP) - 1$		
Carry Flag	Not affected.		
Bytes	1		
Cycles	1		
Example	; Looping with variable stored in stack MOV A, #7 ; Set loop count LOOP_BGN: PUSH A ; Store loop index in stack. ; Operations in loop POP A ; Restore loop index DJNZ A, LOOP_BGN ; Iteration		

ORL A, @DP

Binary Code	<table border="1"><tr><td>0000</td><td>1101</td></tr></table>	0000	1101
0000	1101		
Description	ORL performs the bitwise logical-OR operation between the indirect data memory and ACC. The result is stored in Accumulator.		
Operation	$(A) \leftarrow (A) M[DP]$		
Carry Flag	Not affected.		
Bytes	1		
Cycles	1		
Example	; Assumes M[DP] contains 1 MOV A, #0xA ; Set ACC as 10. ORL A, @DP ; The result, 11 is stored in ACC.		

PUSH A

Binary Code	<table border="1"><tr><td>0000</td><td>0010</td></tr></table>	0000	0010
0000	0010		
Description	The stack pointer is incremented by 1. Then the contents of ACC is copied to the stack.		
Operation	$(SP) \leftarrow (SP) + 1$ $M[SP] \leftarrow (A)$		
Carry Flag	Not affected.		
Bytes	1		
Cycles	1		
Example	PUSH A ; Store ACC in stack MOV A, #0xE ; Assign ACC for port output MOV P2, A ; Drive Port 2 POP A ; Restore ACC from stack		

RET

Binary Code	<table border="1"><tr><td>1001</td><td>0000</td></tr></table>	1001	0000
1001	0000		
Description	Returns from subroutine. The stack pointer is decremented three times.		
Operation	$(PC_{11-8}) \leftarrow M[SP]$ $(SP) \leftarrow (SP) - 1$ $(PC_{7-4}) \leftarrow M[SP]$ $(SP) \leftarrow (SP) - 1$ $(PC_{3-0}) \leftarrow M[SP]$ $(SP) \leftarrow (SP) - 1$		
Carry Flag	Not affected.		
Bytes	1		
Cycles	2		
Example	RET		

SETB C

Binary Code	<table border="1"><tr><td>0000</td><td>0001</td></tr></table>	0000	0001
0000	0001		
Description	Sets the carry flag.		
Operation			
Carry Flag	$(C) \leftarrow 1$		
Bytes	1		
Cycles	1		
Example	SETB C		

RRC A

Binary Code	<table border="1"><tr><td>0000</td><td>1111</td></tr></table>	0000	1111
0000	1111		
Description	Rotates right the contents of ACC with the carry flag.		
Operation	$(A) \leftarrow \{(C), (A_{3-1})\}$		
Carry Flag	$(C) \leftarrow (A_0)$		
Bytes	1		
Cycles	1		
Example	RRC A JC A0_HIGH ; IF $A_0 = 1$ Branches		

SETB @L

Binary Code	<table border="1"><tr><td>1000</td><td>0111</td></tr></table>	1000	0111
1000	0111		
Description	Sets the indirect function flag addressed by DPL.		
Operation	$F[L] \leftarrow 1$		
Carry Flag	Not affected.		
Bytes	1		
Cycles	1		
Example	; Assumes P2 contains 0. MOV L, #1 ; $(L) \leftarrow 1$ SETB @L ; $P2.1 \leftarrow 1$ MOV A, #2 ; $(A) \leftarrow 2$ CJNE A, P2, . ; Wait until P2.1 is 1.		

SETB bit

Binary Code	1000	11bb
Description	Sets a bit in data memory indirectly addressed by DPTR. The bit position is obtained at the least significant two bits of opcode.	
Operation	$M[DP].bit \leftarrow 1$	
Carry Flag	Not affected.	
Bytes	1	
Cycles	1	
Example	; Assumes M[DP] contains 5. SETB 2 ; M[DP].2 \leftarrow 1 CJNE @DP, #7, ERROR ; Check result	

SUB A, @DP

Binary Code	0000	1011
Description	Subtracts the contents of indirect data memory from the Accumulator. The result is stored in Accumulator. The carry flag is cleared if the unsigned value of ACC is less than unsigned value of M[DP]; otherwise, C is set.	
Operation	$(A) \leftarrow (A) - M[DP]$	
Carry Flag	If $(A) < M[DP]$ THEN $(C) \leftarrow 0$ ELSE $(C) \leftarrow 1$.	
Bytes	1	
Cycles	1	
Example	SUB A, @DP	

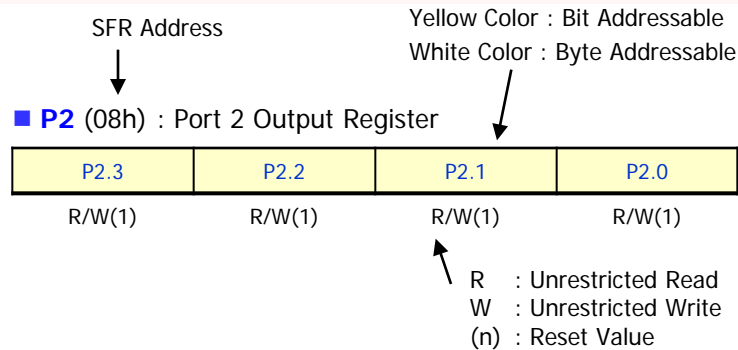
XCH A, @DP

Binary Code	1000	0001
Description	Exchanges the contents of ACC and that of data memory addressed by DPTR.	
Operation	$(A) \leftrightarrow M[DP]$	
Carry Flag	Not affected.	
Bytes	1	
Cycles	1	
Example	XCH A, @DP	

XRL A, @DP

Binary Code	0000	1110
Description	XRL performs the bitwise logical Exclusive-OR operation between the indirect data memory and ACC. The result is stored in Accumulator.	
Operation	$(A) \leftarrow (A) \wedge M[DP]$	
Carry Flag	Not affected.	
Bytes	1	
Cycles	1	
Example	; Assumes M[DP] contains 2 MOV A, #0xA ; Set ACC as 10. XRL A, @DP ; The result, 8 is stored in ACC.	

[How to Read a SFR Descriptions]



■ **P0** (00h) : Port 0 Output Register

P0.3	P0.2	P0.1	P0.0
R/W(1)	R/W(1)	R/W(1)	R/W(1)

■ **P4** (01h) : Port 4 Output Register

P4.3	P4.2	-	-
R/W(1)	R/W(1)		

■ **DPL** (02h) : The Low Nibble of Data Pointer (DPTR)

DPL.3	DPL.2	DPL.1	DPL.0
R/W(0)	R/W(0)	R/W(0)	R/W(0)

■ **DPH** (03h) : The High Nibble of Data Pointer (DPTR)

-	-	DPH.1	DPH.0
		R/W(0)	R/W(0)

■ **P1** (04h) : Port 1 Output Register

P1.3	P1.2	P1.1	P1.0
R/W(1)	R/W(1)	R/W(1)	R/W(1)

■ **REMC** (05h) : The REM Output Control Register

REME	PG2	PG1	PG0
R/W(0)	R/W(0)	R/W(0)	R/W(0)

- ◆ PG[2:0] : Carrier frequency selection.
- ◆ REME : REM output enable.

■ **SPL** (06h) : The Low Nibble of Stack Pointer (SP)

SP.3	SP.2	SP.1	SP.0
R/W(1)	R/W(1)	R/W(1)	R/W(1)

- ◆ Indicate where stack will start.
- ◆ Increment by PUSH and decrement by POP.

■ **SPH** (07h) : The High Nibble of Stack Pointer (SP)

-	-	SPh.1	SPh.0
		R/W(0)	R/W(1)

■ P2 (08h) : Port 2 Output Register

P2.3	P2.2	P2.1	P2.0
R/W(1)	R/W(1)	R/W(1)	R/W(1)

■ IAPCON (09h) : IAP Control Register

RGS1	RGS0	OPS1	OPS0
R/W(0)	R/W(0)	R/W(0)	R/W(0)

◆ RGS[1:0] : Select IAP region.

[0,0] : EEPO (0x1C0 ~ 0x1FF)

[0,1] : EEP1 (0x3C0 ~ 0x3FF)

[1,0] : INFO (0x0 ~ 0x7)

[1,1] : Reserved

◆ OPS[1:0] : Select IAP function.

[0,0] : NO operation

[0,1] : Byte read

[1,0] : Byte erase

[1,1] : Byte write

■ GDL (0Ah) : The Low Nibble of General Purpose Data Register

GDL.3	GDL.2	GDL.1	GDL.0
R/W(0)	R/W(0)	R/W(0)	R/W(0)

■ GDH (0Bh) : The High Nibble of General Purpose Data Register

GDH.3	GDH.2	GDH.1	GDH.0
R/W(0)	R/W(0)	R/W(0)	R/W(0)

■ P3 (0Ch) : Port 3 Output Register

P3.3	P3.2	P3.1	P3.0
R/W(1)	R/W(1)	R/W(1)	R/W(1)

■ CKCFG (0Dh) : The Clock Configuration Register

XT/RG	DIV2	DIV1	DIV0
R/W(0)	R/W(0)	R/W(0)	R/W(0)

◆ XT/RG : System clock source selection.

0 : Internal Ring oscillator is selected as system clock.
External clock oscillator is disabled.

1 : External clock oscillator is selected as system clock.
Internal Ring oscillator is disabled.

Do not set this bit for 8-pin devices.

◆ DIV[2:0] : System clock divider selection.

[0,0,0] : F_{osc}

[0,0,1] : $F_{osc}/2$

[0,1,0] : $F_{osc}/4$

[0,1,1] : $F_{osc}/8$

[1,0,0] : $F_{osc}/16$

[1,0,1] : $F_{osc}/32$

[1,1,0] : $F_{osc}/64$

[1,1,1] : -

■ **IOCFG** (0Eh) : I/O Port Configuration Register

IOMAP1	IOMAP0	P2OEN	-
R/W(0)	R/W(0)	R/W(0)	R/W(0)

- ◆ P2OEN : Configure P2 as push-pull output port.
- ◆ IOMAP [1:0] : Configure I/O ports mapping.
 - [0,0] : Default.
 - [0,1] : Optional 20-pin I/O port mapping
 - [1,0] : Optional 24-pin I/O port mapping
 - [1,1] : Reserved

- ◆ V1.0
 - ✓ Initial Release

- ◆ V1.1
 - ✓ Change Industrial temperature range

- ◆ V1.2
 - ✓ Added ESD information (Page 5)
 - ✓ Added pull-up information (Page 16, 31)