MEGAWIN
MAKE YOU WIN

*8051-Based MCU*

# MG86FE/L104
# Data Sheet

**Version: A1.5**

## Features

- 1-T 80C51 Central Processing Unit
- **MG86FE/L104** with 4K Bytes flash ROM
  - ISP memory zone could be optioned as 0.5KB/1KB/1.5KB……3.5KB
  - Flexible IAP size.
  - Code protection for flash memory access
  - Flash write/erase cycle
    - For 0.5K IAP, the MTP of IAP write cycle is 2,000 times.
    - For 1.0K IAP, the MTP of IAP write cycle is 1,000 times
  - Flash data retention: 100 years at 25℃
  - MG86FE/L104 Flash space mapping (Default)
    - AP Flash（0000h~07FFh）
    - IAP Flash（0800h~0BFFh）
    - ISP Flash（0C00h~0FFFh）（ISP Boot code）
- On-chip 256 bytes scratch-pad RAM
- Interrupt controller
  - 6 sources, four-level-priority interrupt capability
  - Two external interrupt inputs, nINT0 and nINT1.
  - All external interrupts support High/Low level or Rising/Falling edge trigger.
- Two 16-bit timer/counters, Timer 0 and Timer 1.
  - T0CKO on P34 and T1CKO on P35.
  - X12 mode enabled for T0/T1.
- Enhanced UART (S0)
  - Framing Error Detection
  - Automatic Address Recognition
  - a speed improvement mechanism (X2/X4 mode)
  - 4 channel UART repeater mechanism
- Keypad Interrupt on all GPIO.
- Programmable Watchdog Timer
  - One time enabled by CPU or power-on
  - Sourced from on-chip low frequency oscillator (ILRCO)
  - Interrupt CPU or Reset CPU on WDT overflow
  - Support WDT function in power down mode
- Maximum 18 GPIOs in 20-pin package.
  - P3 can be configured to quasi-bidirectional, push-pull output, open-drain output and input only.
  - P1, P4.0 and P4.1 can be configured to push-pull output or open-drain output.
  - P4.0, P4.1 and P3.6 are shared with XTAL2, XTAL1 and RST.
  - All GPIOs have wakeup capability.
- Multiple power control modes: idle mode, power-down mode, slow mode, sub-clock mode, watch mode and monitor mode.
  - All interrupts can wake up IDLE mode
  - 6 sources to wake up Power-Down mode
  - Slow mode and sub-clock mode support low speed MCU operation
  - Watch mode supports WDT to resume CPU in power down
  - Monitor mode supports BOD0 to resume CPU in power down (L-series only)
- Brown-Out Detector: VDD 4.2V for E-series and VDD 2.4V for L-series
  - Interrupt CPU or reset CPU
  - Wake up CPU in Power-Down mode (L-series only)

- Operating voltage:
  - MG86FE104: 4.2V~5.5V, minimum 4.5V requirement in flash write operation (ISP/IAP/ICP)
  - MG86FL104: 2.4V~3.6V, minimum 2.7V requirement in flash write operation (ISP/IAP/ICP)
- Operating frequency range: 25MHz(max)
  - MG86FE104: 0 – 12MHz @ 4.2V – 5.5V and 0 – 25MHz @ 4.5V – 5.5V
  - MG86FL104: 0 – 12MHz @ 2.4V – 3.6V and 0 – 25MHz @ 2.7V – 3.6V
- Clock Source:
  - Internal 24MHz/22.118MHz oscillator (IHRCO): factory calibrated to ±1%, typical
  - External crystal mode
  - Internal Low frequency RC Oscillator (ILRCO) support (about 64KHz)
  - External clock input (ECKI) on XTAL2/P4.0
  - Internal Oscillator output on XTAL2/P4.0
- Operating Temperature:
  - Industrial (-40℃ to +85℃)*
- Package Types:
  - SOP20: MG86FE/L104AS20
  - SOP16: MG86FE/L104AS16
  - SOP8: MG86FE/L104AS8

*: Tested by sampling.

# Content

# 1. General Description

The **MG86FE/L104** is a single-chip microcontroller based on a high performance 1-T architecture 80C51 CPU that executes instructions in 1~6 clock cycles (about 6~7 times the rate of a standard 8051 device), and has an 8051 compatible instruction set. Therefore at the same performance as the standard 8051, the **MG86FE/L104** can operate at a much lower speed and thereby greatly reduce the power consumption.

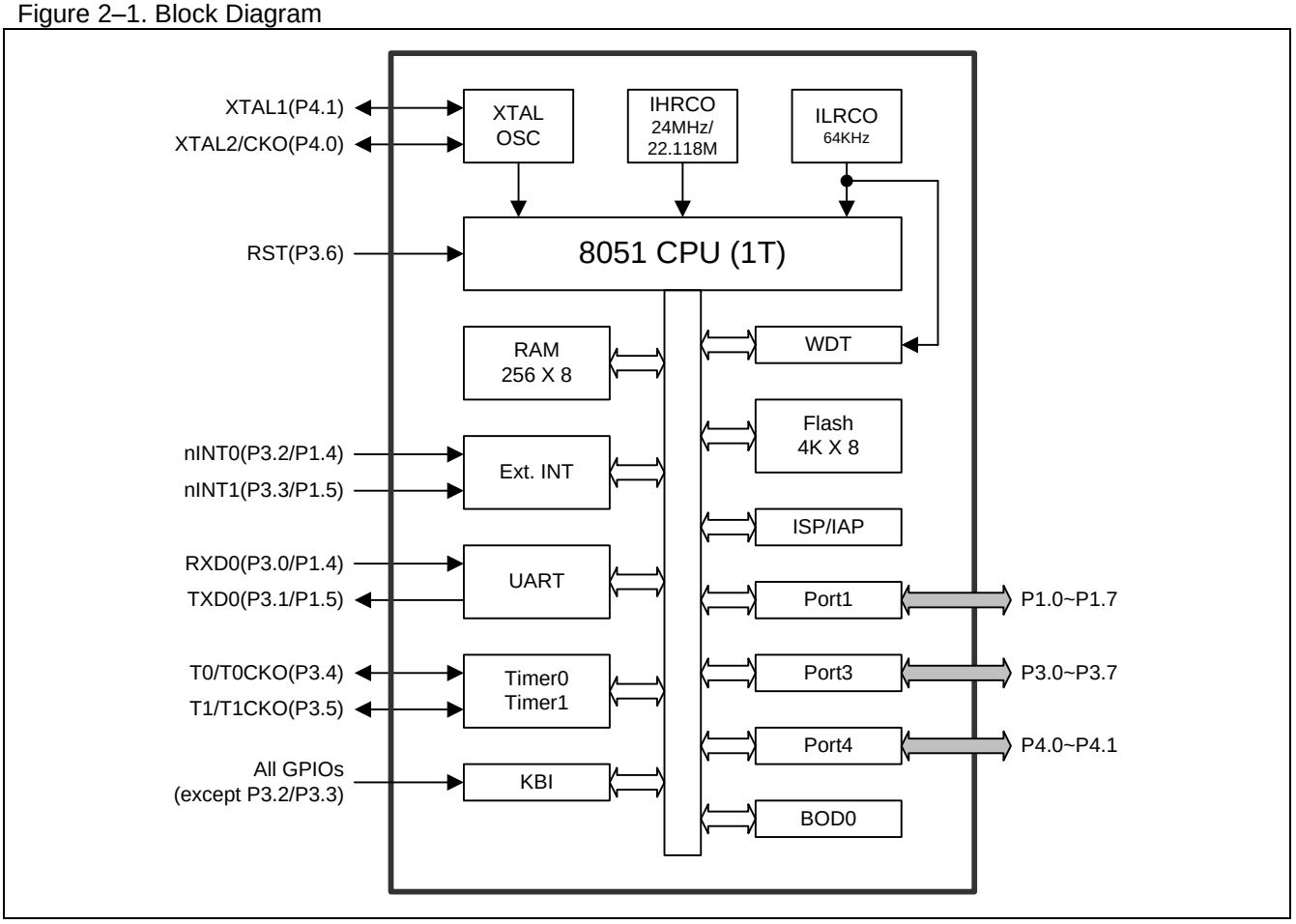
The **MG86FE/L104** has 4K bytes of embedded Flash memory for code and data. The Flash memory can be programmed either in serial writer mode (via ICP, In-Circuit-Programming) or in ISP (In-System Programming) mode. And, it also provides the In-Application Programming (IAP) capability. ICP and ISP allow the user to download new code without removing the microcontroller from the actual end product; IAP means that the device can write non-volatile data in the Flash memory while the application program is running. There needs no external high voltage for programming due to its built-in charge-pumping circuitry.

The **MG86FE/L104** retains all features of the standard 80C52 with 256 bytes of scratch-pad RAM, two 8bit I/O ports, two external interrupts with High/Low trigger option, a multi-source 4-level interrupt controller and two 16-bits timer/counters. In addition, the **MG86FE/L104** has two extra I/O pins (P4.0 and P4.1), keypad interrupt, an one-time enabled Watchdog Timer, a Brown-out Detector, an on-chip crystal oscillator (shared with P4.0 and P4.1), a high precision internal oscillator (IHRCO), an internal low speed RC oscillator (ILRCO) and a more versatile serial channel that facilitates multiprocessor communication (EUART) and a speed improvement mechanism (X2/X4 mode).

The **MG86FE/L104** has multiple operating modes to reduce the power consumption: idle mode, power down mode, slow mode, sub-clock mode, watch mode and monitor mode. In the Idle mode the CPU is frozen while the peripherals and the interrupt system are still operating. In the Power-Down mode the RAM and SFRs' value are saved and all other functions are inoperative; most importantly, in the Power-down mode the device can be waked up by many interrupt or reset sources. In slow mode, the user can further reduce the power consumption by using the 8-bit system clock pre-scaler to slow down the operating speed. Or select sub-clock mode which clock source is derived from internal low speed oscillator (ILRCO) for CPU to perform an ultra low speed operation. In watch mode, it keeps WDT running in power-down or idle mode and resumes CPU when WDT overflows. Monitor mode provides the Brown-Out detection in power down mode and resumes CPU when chip VDD reaches the specific detection level.

# 2. Block Diagram

Figure 2–1. Block Diagram



# 3. Special Function Register

## 3.1. SFR Map

Table 3–1. SFR Map

| | 0/8 | 1/9 | 2/A | 3/B | 4/C | 5/D | 6/E | 7/F |
|---|---|---|---|---|---|---|---|---|
| **F8** | -- | -- | -- | -- | -- | -- | -- | -- |
| **F0** | B | -- | -- | -- | -- | -- | -- | -- |
| **E8** | P4 | -- | -- | -- | -- | -- | -- | -- |
| **E0** | ACC | WDTCR | IFD | IFADRH | IFADRL | IFMT | SCMD | ISPCR |
| **D8** | -- | -- | -- | -- | -- | -- | -- | -- |
| **D0** | PSW | -- | -- | -- | -- | -- | P3KBIE | P1KBIE |
| **C8** | -- | -- | -- | -- | -- | -- | -- | -- |
| **C0** | -- | -- | -- | -- | -- | -- | -- | CKCON0 |
| **B8** | IP0L | SADEN | -- | -- | -- | -- | -- | |
| **B0** | P3 | P3M0 | P3M1 | P4M0 | PUCON0 | | | IP0H |
| **A8** | IE | SADDR | -- | -- | | EIE1 | EIP1L | EIP1H |
| **A0** | -- | AUXR0 | AUXR1 | AUXR2 | -- | -- | -- | -- |
| **98** | SCON | SBUF | -- | -- | -- | -- | | |
| **90** | P1 | P1M0 | -- | -- | -- | -- | -- | PCON1 |
| **88** | TCON | TMOD | TL0 | TL1 | TH0 | TH1 | SFIE | -- |
| **80** | -- | SP | DPL | DPH | -- | -- | -- | PCON0 |
| | **0/8** | **1/9** | **2/A** | **3/B** | **4/C** | **5/D** | **6/E** | **7/F** |

## 3.2. SFR Bit Assignment

Table 3–2. SFR Bit Assignment

| SYMBOL | DESCRIPTION | ADDR | BIT ADDRESS AND SYMBOL | | | | | | | | RESET VALUE |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | Bit-7 | Bit-6 | Bit-5 | Bit-4 | Bit-3 | Bit-2 | Bit-1 | Bit-0 | |
| SP | Stack Pointer | 81H | | | | | | | | | 00000111B |
| DPL | Data Pointer Low | 82H | | | | | | | | | 00000000B |
| DPH | Data Pointer High | 83H | | | | | | | | | 00000000B |
| PCON0 | Power Control 0 | 87H | SMOD1 | SMOD0 | GF | POF | GF1 | GF0 | PD | IDL | 00010000B |
| *TCON* | *Timer Control* | *88H* | *TF1* | *TR1* | *TF0* | *TR0* | *IE1* | *IT1* | *IE0* | *IT0* | *00000000B* |
| TMOD | Timer Mode | 89H | GATE | C/T | M1 | M0 | GATE | C/T | M1 | M0 | 00000000B |
| TL0 | Timer Low 0 | 8AH | | | | | | | | | 00000000B |
| TL1 | Timer Low 1 | 8BH | | | | | | | | | 00000000B |
| TH0 | Timer High 0 | 8CH | | | | | | | | | 00000000B |
| TH1 | Timer High 1 | 8DH | | | | | | | | | 00000000B |
| SFIE | System Flag INT En. | 8EH | UTIE | -- | -- | -- | KBIFIE | -- | BOF0IE | WDTFIE | 0xxx0x00B |
| *P1* | *Port 1* | *90H* | *P1.7* | *P1.6* | *P1.5* | *P1.4* | *P1.3* | *P1.2* | *P1.1* | *P1.0* | *11111111B* |
| P1M0 | P1 Mode Register 0 | 91H | P1M0.7 | P1M0.6 | P1M0.5 | P1M0.4 | P1M0.3 | P1M0.2 | P1M0.1 | P1M0.0 | 00000000B |
| PCON1 | Power Control 1 | 97H | SWRF | EXRF | -- | -- | KBIF | -- | BOF0 | WDTF | 00100x00B |
| *SCON* | *Serial Control* | *98H* | *SM0/FE* | *SM1* | *SM2* | *REN* | *TB8* | *RB8* | *Ti* | *RI* | *00000000B* |
| SBUF | Serial Buffer | 99H | | | | | | | | | xxxxxxxxB |
| AUXR0 | Auxiliary Register 0 | A1H | P40OC1 | P40OC0 | P40FD | GF | P1FS1 | P1FS0 | INT1H | INT0H | 00000000B |
| AUXR1 | Auxiliary Register 1 | A2H | RTX3E | RTX2E | RTX1E | RTX0E | GF | GF | RXCS1 | RXCS0 | 00000000B |
| AUXR2 | Auxiliary Register 2 | A3H | URXR | BTI | URM0X6 | SMOD2 | T1X12 | T0X12 | T1CKOE | T0CKOE | 00000000B |
| *IE* | *Interrupt Enable* | *A8H* | *EA* | *--* | *--* | *ES* | *ET1* | *EX1* | *ET0* | *EX0* | *00000000B* |
| SADDR | Slave Address | A9H | | | | | | | | | 00000000B |
| EIE1 | Extended INT Enable 1 | ADH | -- | -- | -- | -- | ESF | -- | -- | -- | xxxx0xxxB |
| EIP1L | Ext. INT Priority 1 Low | AEH | -- | -- | -- | -- | PSFL | -- | -- | -- | xxxx0xxxB |
| EIP1H | Ext. INT Priority 1 High | AFH | -- | -- | -- | -- | PSFH | -- | -- | -- | xxxx0xxxB |
| *P3* | *Port 3* | *B0H* | *P3.7* | *P3.6* | *P3.5* | *P3.4* | *P3.3* | *P3.2* | *P3.1* | *P3.0* | *11111111B* |
| P3M0 | P3 Mode Register 0 | B1H | P3M0.7 | P3M0.6 | P3M0.5 | P3M0.4 | P3M0.3 | P3M0.2 | P3M0.1 | P3M0.0 | 00000000B |
| P3M1 | P3 Mode Register 1 | B2H | P3M1.7 | P3M1.6 | P3M1.5 | P3M1.4 | P3M1.3 | P3M1.2 | P3M1.1 | P3M1.0 | 00000000B |
| P4M0 | P4 Mode Register 0 | B3H | -- | -- | -- | -- | -- | -- | P4M0.1 | P4M0.0 | xxxxxx00B |
| PUCON0 | Pull-Up Control 0 | 92H | -- | PU40 | -- | -- | PU11 | PU10 | -- | -- | x0xx00xxB |
| IP0H | Interrupt Priority 0 High | B7H | -- | -- | -- | PSH | PT1H | PX1H | PT0H | PX0H | 00000000B |
| *IP0L* | *Interrupt Priority Low* | *B8H* | *--* | *--* | *--* | *PSL* | *PT1L* | *PX1L* | *PT0L* | *PX0L* | *00000000B* |
| SADEN | Slave Address Mask | B9H | | | | | | | | | 00000000B |
| CKCON0 | Clock Control 0 | C7H | AFS | -- | -- | -- | -- | SCKS2 | SCKS1 | SCKS0 | 0xxxx000B |
| *PSW* | *Program Status Word* | *D0H* | *CY* | *AC* | *F0* | *RS1* | *RS0* | *OV* | *F1* | *P* | *00000000B* |
| P3KBIE | P3 KBI Enable | D6H | P37KBIE | P36KBIE | P35KBIE | P34KBIE | P41KBIE | P40KBIE | P31KBIE | P30KBIE | 00000000B |
| P1KBIE | P1 KBI Enable | D7H | P17KBIE | P16KBIE | P15KBIE | P14KBIE | P13KBIE | P12KBIE | P11KBIE | P10KBIE | 00000000B |
| *ACC* | *Accumulator* | *E0H* | *ACC.7* | *ACC.6* | *ACC.5* | *ACC.4* | *ACC.3* | *ACC.2* | *ACC.1* | *ACC.0* | *00000000B* |
| WDTCR | Watch-dog-timer Control register | E1H | WREN | NSW | ENW | CLW | WIDL | PS2 | PS1 | PS0 | 0x000000B |
| IFD | ISP Flash data | E2H | | | | | | | | | 11111111B |
| IFADRH | ISP Flash address High | E3H | | | | | | | | | 00000000B |
| IFADRL | ISP Flash Address Low | E4H | | | | | | | | | 00000000B |
| IFMT | ISP Mode Table | E5H | -- | -- | -- | -- | -- | MS2 | MS1 | MS0 | xxxxx000B |
| SCMD | ISP Serial Command | E6H | | | | | | | | | xxxxxxxxB |
| ISPCR | ISP Control Register | E7H | ISPEN | BS | SRST | CFAIL | -- | -- | -- | -- | 0000xxxxB |
| *P4* | *Port 4* | *E8H* | *--* | *--* | *--* | *--* | *--* | *--* | *P4.1* | *P4.0* | *xxxxxx11B* |
| *B* | *B Register* | *F0H* | *B.7* | *B.6* | *B.5* | *B.4* | *B.3* | *B.2* | *B.1* | *B.0* | *00000000B* |

## 3.3. Auxiliary SFR Map (Page P)

**MG86FE/L104** has an auxiliary SFR page which is indexed by page P and the SFRs' write is a different way from standard 8051 SFR page. The registers in auxiliary SFR map are addressed by IFMT and SCMD like ISP/IAP access flow. Page P has 256 bytes space that can target to **5** physical bytes and **4** logical bytes. The **5** physical bytes include IAPLB, CKCON2, PCON2, SPCON0 and DCON0. The **4** logical bytes include PCON0, PCON1, CKCON0 and WDTCR. Write on the **4** logical bytes gets the coherence content with the same SFR in Normal Page. Please refer Section "18 Page P SFR Access" for more detail information.

Table 3–3. Auxiliary SFR Map (Page P)

| | | 0/8 | 1/9 | 2/A | 3/B | 4/C | 5/D | 6/E | 7/F |
|---|---|---|---|---|---|---|---|---|---|
| **F8** | | -- | | | | | | | |
| **F0** | | | | | | | | | |
| **E8** | | -- | | | | | | | |
| **E0** | | | WDTCR | | | | | | |
| **D8** | | | | | | | | | |
| **D0** | | | | | | | | | |
| **C8** | | | | | | | | | |
| **C0** | | | | | | | | | CKCON0 |
| **B8** | | | | | | | | | -- |
| **B0** | | | | | | | | | |
| **A8** | | | | | | | | | |
| **A0** | | | | | | | | | |
| **98** | | | | | | | | | |
| **90** | | | | | | | | | PCON1 |
| **88** | | | | | | | | | |
| **80** | | | | | | | | | PCON0 |
| **78** | | | | | | | | | |
| **70** | | | | | | | | | |
| **68** | | | | | | | | | |
| **60** | | | | | | | | | |
| **58** | | | | | | | | | |
| **50** | | | | | | | | | |
| **48** | | SPCON0 | | | | DCON0 | | | |
| **40** | | CKCON2 | -- | | | PCON2 | -- | | |
| **38** | | | | | | | | | |
| **30** | | | | | | | | | |
| **28** | | | | | | | | | |
| **20** | | | | | | | | | |
| **18** | | | | | | | | | |
| **10** | | | | | | | | | |
| **08** | | | | | | | | | |
| **00** | | -- | -- | -- | IAPLB | -- | -- | -- | -- |
| | | **0/8** | **1/9** | **2/A** | **3/B** | **4/C** | **5/D** | **6/E** | **7/F** |

## 3.4. Auxiliary SFR Bit Assignment (Page P)

Table 3–4. Auxiliary SFR Bit Assignment (Page P)

| SYMBOL | DESCRIPTION | ADDR | BIT ADDRESS AND SYMBOL | | | | | | | | RESET VALUE |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | Bit-7 | Bit-6 | Bit-5 | Bit-4 | Bit-3 | Bit-2 | Bit-1 | Bit-0 | |
| IAPLB | IAP Low Boundary | 03H | IAPLB6 | IAPLB5 | IAPLB4 | IAPLB3 | IAPLB2 | IAPLB1 | IAPLB0 | -- | IFR03 |
| CKCON2 | Clock Control 2 | 40H | 0 | 0 | XTALE | IHRCOE | 0 | 0 | OSCS1 | OSCS0 | xx01xx00B |
| PCON2 | Power Control 2 | 44H | -- | AWBOD0 | -- | -- | -- | -- | BO0RE | **1** | x0xxxxx1B |
| SPCON0 | SFR Page Control 0 | 48H | -- | -- | -- | WRCTL | -- | CKCTL0 | PWCTL1 | PWCTL0 | xxx0x000B |
| DCON0 | Device Control 0 | 4CH | HSE | IAPO | 0 | 0 | 0 | 0 | RSTIO | 0 | 10xxxxxx |

Sample Code of Page-P SFR write：

```
IFADRH = 0x00;
ISPCR = ISPEN;                  //enable IAP/ISP
IFMT = MS2;                     // Page-P write，IFMT =0x04
IFADRL = SPCON0;                //Set Page-P SFR address
IFD |= CKCTL0;                  // set CKCTL0
SCMD = 0x46;                    //
SCMD = 0xB9;                    //
IFMT = Flash_Standby;           // IAP/ISP standby，IFMT =0x00
ISPCR &= ~ISPEN;
```

# 4. Pin Configurations

## 4.1. Package Instruction

Figure 4–1. MG86FE/L104AS20 Top View

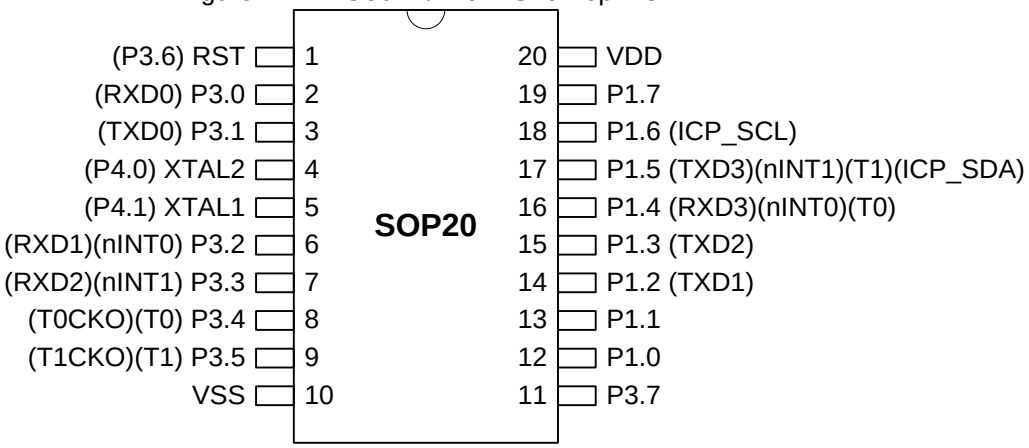



Figure 4–2. MG86FE/L104AS16 Top View

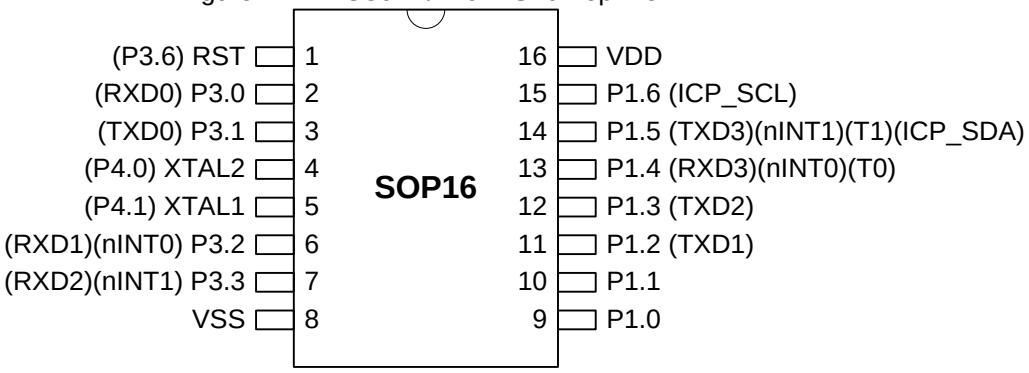
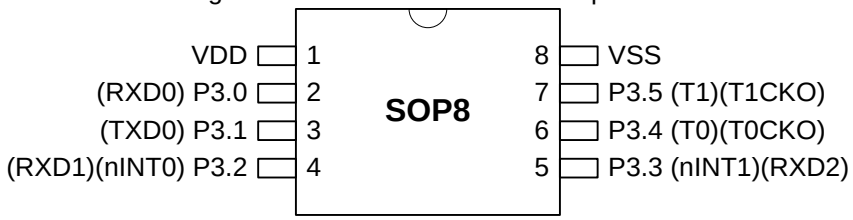



Figure 4–3. MG86FE/L104AS8 Top View



## 4.2. Pin Description

Table 4–1. Pin Description

| MNEMONIC | PIN NUMBER | | | I/O TYPE | DESCRIPTION |
|---|---|---|---|---|---|
| | 20-Pin SOP | 16-Pin SOP | 8-Pin SOP | | |
| P1.0 | 12 | 9 | -- | I/O | * Port 1.0. |
| P1.1 | 13 | 10 | -- | I/O | * Port 1.1. |
| P1.2<br>(TXD1) | 14 | 11 | -- | I/O | * Port 1.2.<br>* TXD1: TXD channel 1 output of UART repeater. |
| P1.3<br>(TXD2) | 15 | 12 | -- | I/O | * Port 1.3.<br>* TXD2: TXD channel 2 output of UART repeater. |
| P1.4<br>(RXD3)<br>(nINT0)<br>(T0) | 16 | 13 | -- | I/O | * Port 1.4.<br>* RXD3: RXD channel 3 input of UART repeater.<br>* nINT0: Alternate function for nINT0 input.<br>* T0: Alternate function for T0 input. |
| P1.5<br>(TXD3)<br>(nINT1)<br>(T1)<br>**(ICP_SDA)** | 17 | 14 | -- | I/O | * Port 1.5.<br>* TXD3: TXD channel 3 output of UART repeater.<br>* nINT1: Alternate function for nINT1 input.<br>* T1: Alternate function for T1 input.<br>* ICP_SDA: Serial data of ICP interface. |
| P1.6<br>**(ICP_SCL)** | 18 | 15 | -- | I/O | * Port 1.6.<br>* ICP_SCL: Serial clock of ICP interface. |
| P1.7 | 19 | -- | -- | I/O | * Port 1.7. |
| P3.0<br>(RXD0) | 2 | 2 | 2 | I/O | * Port 3.0.<br>* RXD0: UART0 serial input port. |
| P3.1<br>(TXD0) | 3 | 3 | 3 | I/O | * Port 3.1.<br>* TXD0: UART0 serial output port. |
| P3.2<br>(nINT0)<br>(RXD1) | 6 | 6 | 4 | I/O | * Port 3.2.<br>* nINT0: external interrupt 0 input.<br>* RXD3: RXD channel 3 input of UART repeater. |
| P3.3<br>(nINT1)<br>(RXD2) | 7 | 7 | 5 | I/O | * Port 3.3.<br>* nINT1: external interrupt 1 input.<br>* RXD3: RXD channel 3 input of UART repeater. |
| P3.4<br>(T0)<br>(T0CKO) | 8 | -- | 6 | I/O | * Port 3.4.<br>* T0: Timer/Counter 0 external input.<br>* T0CKO: programmable clock-out from Timer 0. |
| P3.5<br>(T1)<br>(T1CKO) | 9 | -- | 7 | I/O | * Port 3.5.<br>* T1: Timer/Counter 1 external input.<br>* T1CKO: programmable clock-out from Timer 1. |
| P3.7 | 11 | -- | -- | I/O | * Port 3 bit-7. |
| P4.0<br>(CKO)<br>(ECKI)<br>(XTAL2) | 4 | 4 | -- | I/O<br>O<br>I<br>O | * Port 4.0. It is only accessed in SFR page "**1**".<br>* CKO: Internal clock output.<br>* ECKI: In external clock input mode, this is clock input pin.<br>*XTAL2: Output of on-chip crystal oscillating circuit. |
| P4.1<br>(XTAL1) | 5 | 5 | -- | I/O<br>I | * Port 4.1. It is only accessed in SFR page "**1**".<br>* XTAL1: Input of on-chip crystal oscillating circuit. |
| RST<br>(P3.6) | 1 | 1 | -- | I<br>I/O | * RST: External RESET input, high active.<br>* Port 3.6. |
| VDD | 20 | 16 | 1 | P | Power supply input. 5V input for E-type. 3.3V input for L-type. |
| VSS | 10 | 8 | 8 | G | Ground, 0 V reference. |

# 5. 8051 CPU Function Description

## 5.1. CPU Register

***PSW: Program Status Word***

SFR Page = Normal
SFR Address = 0xD0 RESET = 0000-0000

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| CY | AC | F0 | RS1 | RS0 | OV | F1 | P |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

CY: Carry bit.
AC: Auxiliary carry bit.
F0: General purpose flag 0.
RS1: Register bank select bit 1.
RS0: Register bank select bit 0.
OV: Overflow flag.
F1: General purpose flag 1.
P: Parity bit.

The program status word(PSW) contains several status bits that reflect the current state of the CPU. The PSW, shown above, resides in the SFR space. It contains the Carry bit, the Auxiliary Carry(for BCD operation), the two register bank select bits, the Overflow flag, a Parity bit and two user-definable status flags.

The Carry bit, other than serving the function of a Carry bit in arithmetic operations, also serves as the "Accumulator" for a number of Boolean operations.

The bits RS0 and RS1 are used to select one of the four register banks shown in Section "6.2 On-Chip Data RAM". A number of instructions refer to these RAM locations as R0 through R7.

The Parity bit reflects the number of 1s in the Accumulator. P=1 if the Accumulator contains an odd number of 1s and otherwise P=0.

***SP: Stack Pointer***

SFR Page = Normal
SFR Address = 0x81 RESET = 0000-0111

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| SP.7 | SP.6 | SP.5 | SP.4 | SP.3 | SP.2 | SP.1 | SP.0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

The Stack Pointer holds the location of the top of the stack. The stack pointer is incremented before every PUSH operation. The SP register defaults to 0x07 after reset.

***DPL: Data Pointer Low***

SFR Page = Normal
SFR Address = 0x82 RESET = 0000-0000

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| DPL.7 | DPL.6 | DPL.5 | DPL.4 | DPL.3 | DPL.2 | DPL.1 | DPL.0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

The DPL register is the low byte of the 16-bit DPTR. DPTR is used to access indirectly addressed XRAM and Flash memory.

***DPH: Data Pointer High***

SFR Page = Normal
SFR Address = 0x83 RESET = 0000-0000

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| DPH.7 | DPH.6 | DPH.5 | DPH.4 | DPH.3 | DPH.2 | DPH.1 | DPH.0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

The DPH register is the high byte of the 16-bit DPTR. DPTR is used to access indirectly addressed XRAM and Flash memory.

***ACC: Accumulator***

SFR Page = Normal
SFR Address = 0xE0 RESET = 0000-0000

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| ACC.7 | ACC.6 | ACC.5 | ACC.4 | ACC.3 | ACC.2 | ACC.1 | ACC.0 |
| **R/W** | **R/W** | **R/W** | **R/W** | **R/W** | **R/W** | **R/W** | **R/W** |

This register is the accumulator for arithmetic operations.

***B: B Register***

SFR Page = Normal
SFR Address = 0xF0 RESET = 0000-0000

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| B.7 | B.6 | B.5 | B.4 | B.3 | B.2 | B.1 | B.0 |
| **R/W** | **R/W** | **R/W** | **R/W** | **R/W** | **R/W** | **R/W** | **R/W** |

This register serves as a second accumulator for certain arithmetic operations.

## 5.2. CPU Timing

The **MG86FE/L104** is a single-chip microcontroller based on a high performance 1-T architecture 80C51 CPU that has an 8051 compatible instruction set, and executes instructions in 1~6 clock cycles (about 6~7 times the rate of a standard 8051 device). It employs a pipelined architecture that greatly increases its instruction throughput over the standard 8051 architecture. The instruction timing is different than that of the standard 8051.

In many 8051 implementations, a distinction is made between machine cycles and clock cycles, with machine cycles varying from 2 to 12 clock cycles in length. However, the 1T-80C51 implementation is based solely on clock cycle timing. All instruction timings are specified in terms of clock cycles. For more detailed information about the 1T-80C51 instructions, please refer Section "23 Instruction Set" which includes the mnemonic, number of bytes, and number of clock cycles for each instruction.

## 5.3. CPU Addressing Mode

***Direct Addressing(DIR)***
In direct addressing the operand is specified by an 8-bit address field in the instruction. Only internal data RAM and SFRs can be direct addressed.

***Indirect Addressing(IND)***
In indirect addressing the instruction specified a register which contains the address of the operand. Both internal and external RAM can be indirectly addressed.

The address register for 8-bit addresses can be R0 or R1 of the selected bank, or the Stack Pointer.
The address register for 16-bit addresses can only be the 16-bit data pointer register – DPTR.

***Register Instruction(REG)***
The register banks, containing registers R0 through R7, can be accessed by certain instructions which carry a 3-bit register specification within the op-code of the instruction. Instructions that access the registers this way are code efficient because this mode eliminates the need of an extra address byte. When such instruction is executed, one of the eight registers in the selected bank is accessed.

***Register-Specific Instruction***
Some instructions are specific to a certain register. For example, some instructions always operate on the accumulator or data pointer, etc. No address byte is needed for such instructions. The op-code itself does it.

***Immediate Constant(IMM)***
The value of a constant can follow the op-code in the program memory.

***Index Addressing***
Only program memory can be accessed with indexed addressing and it can only be read. This addressing mode is intended for reading look-up tables in program memory. A 16-bit base register(either DPTR or PC) points to the base of the table, and the accumulator is set up with the table entry number. Another type of indexed addressing is used in the conditional jump instruction.
In conditional jump, the destination address is computed as the sum of the base pointer and the accumulator.

# 6. Memory Organization

Like all 80C51 devices, the **MG86FE/L104** has separate address spaces for program and data memory. The logical separation of program and data memory allows the data memory to be accessed by 8-bit addresses, which can be quickly stored and manipulated by the 8-bit CPU.

Program memory (ROM) can only be read, not written to. There can be up to 4K bytes of program memory. In the **MG86FE/L104**, all the program memory are on-chip Flash memory, and without the capability of accessing external program memory because of no External Access Enable (/EA) and Program Store Enable (/PSEN) signals designed.

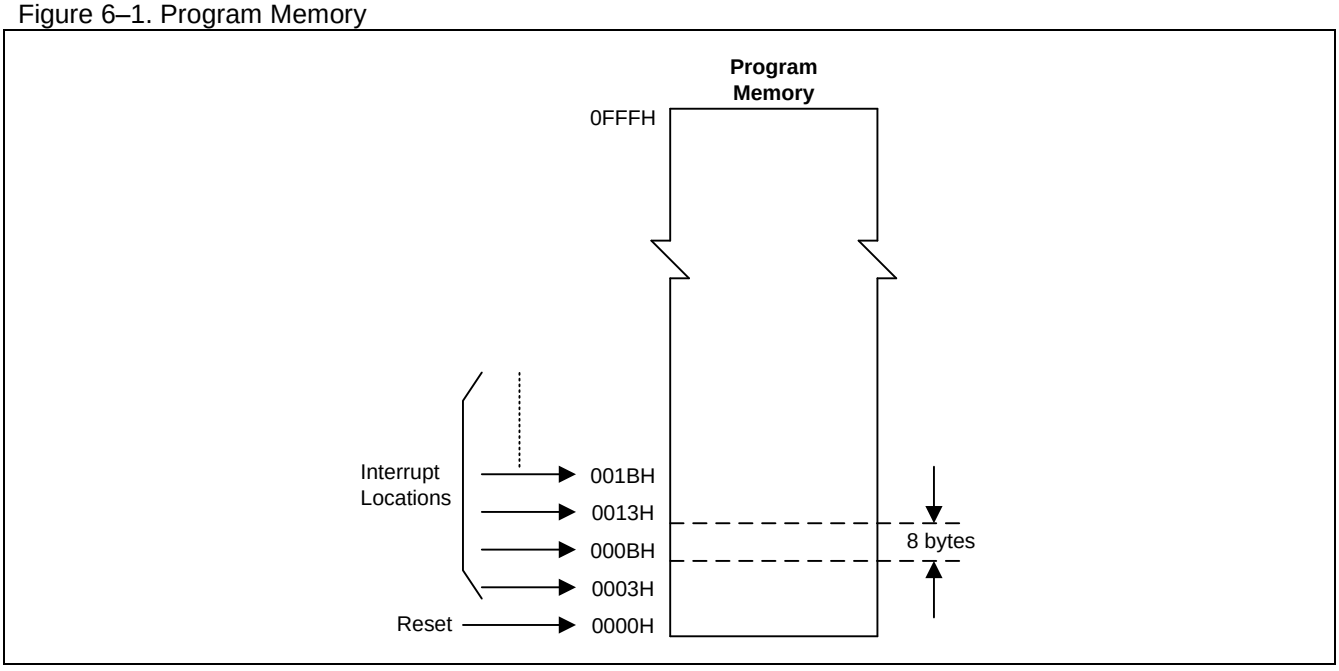
Data memory occupies a separate address space from program memory. In the **MG86FE/L104**, there is only a 256 bytes of internal scratch-pad RAM and has no any expanded RAM(XRAM).

## 6.1. On-Chip Program Flash

Program memory is the memory which stores the program codes for the CPU to execute, as shown in Figure 6–1. After reset, the CPU begins execution from location 0000H, where should be the starting of the user's application code. To service the interrupts, the interrupt service locations (called interrupt vectors) should be located in the program memory. Each interrupt is assigned a fixed location in the program memory. The interrupt causes the CPU to jump to that location, where it commences execution of the service routine. External Interrupt 0, for example, is assigned to location 0003H. If External Interrupt 0 is going to be used, its service routine must begin at location 0003H. If the interrupt is not going to be used, its service location is available as general purpose program memory.

The interrupt service locations are spaced at an interval of 8 bytes: 0003H for External Interrupt 0, 000BH for Timer 0, 0013H for External Interrupt 1, 001BH for Timer 1, etc. If an interrupt service routine is short enough (as is often the case in control applications), it can reside entirely within that 8-byte interval. Longer service routines can use a jump instruction to skip over subsequent interrupt locations, if other interrupts are in use.

Figure 6–1. Program Memory



## 6.2. On-Chip Data RAM

Figure 6–2 shows the internal and external data memory spaces available to the **MG86FE/L104** user. Internal data memory can be divided into three blocks, which are generally referred to as the lower 128 bytes of RAM, the upper 128 bytes of RAM, and the 128 bytes of SFR space. Internal data memory addresses are always 8-bit wide, which implies an address space of only 256 bytes. Direct addresses higher than 7FH access the SFR space; and indirect addresses higher than 7FH access the upper 128 bytes of RAM. Thus the SFR space and the upper 128 bytes of RAM occupy the same block of addresses, 80H through FFH, although they are physically separate entities.

The lower 128 bytes of RAM are present in all 80C51 devices as mapped in Figure 6–3. The lowest 32 bytes are grouped into 4 banks of 8 registers. Program instructions call out these registers as R0 through R7. Two bits in the Program Status Word (PSW) select which register bank is in use. This allows more efficient use of code space, since register instructions are shorter than instructions that use direct addressing. The next 16 bytes above the register banks form a block of bit-addressable memory space. The 80C51 instruction set includes a wide selection of single-bit instructions, and the 128 bits in this area can be directly addressed by these instructions. The bit addresses in this area are 00H through 7FH.

All of the bytes in the Lower 128 can be accessed by either direct or indirect addressing while the Upper 128 can only be accessed by indirect addressing.

Figure 6–4 gives a brief look at the Special Function Register (SFR) space. SFRs include the Port latches, timers, peripheral controls, etc. These registers can only be accessed by direct addressing. Sixteen addresses in SFR space are both byte- and bit-addressable. The bit-addressable SFRs are those whose address ends in 0H or 8H.

Figure 6–2. Data Memory

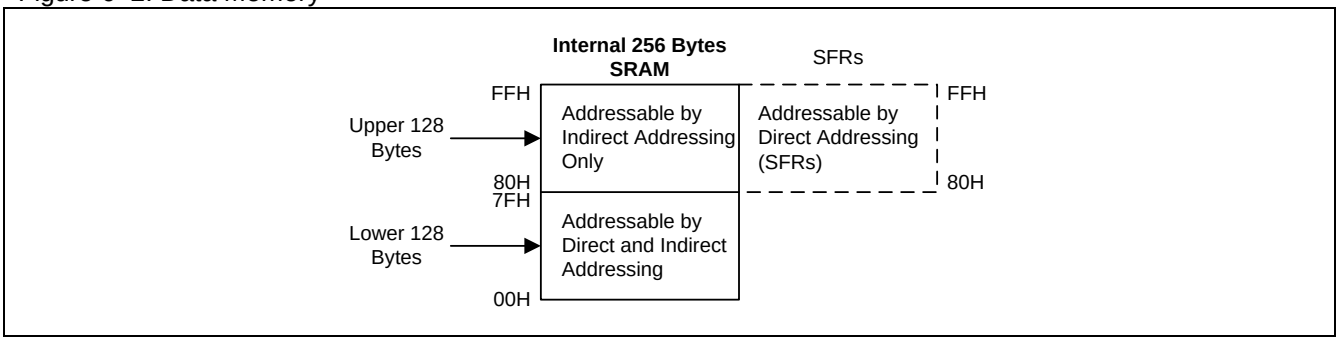



Figure 6–3. Lower 128 Bytes of Internal RAM

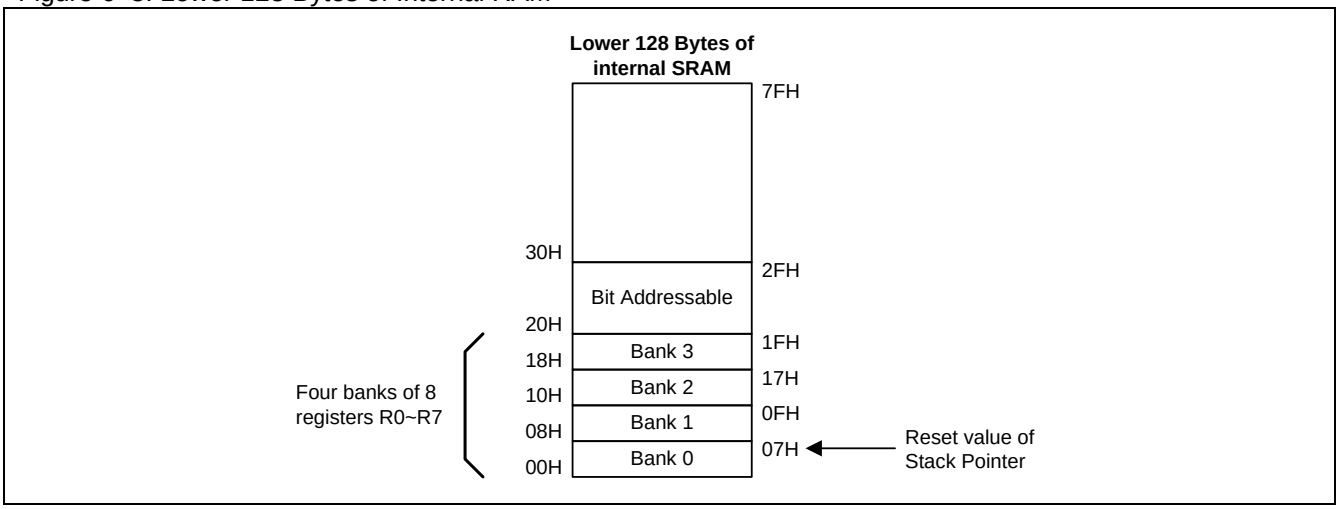
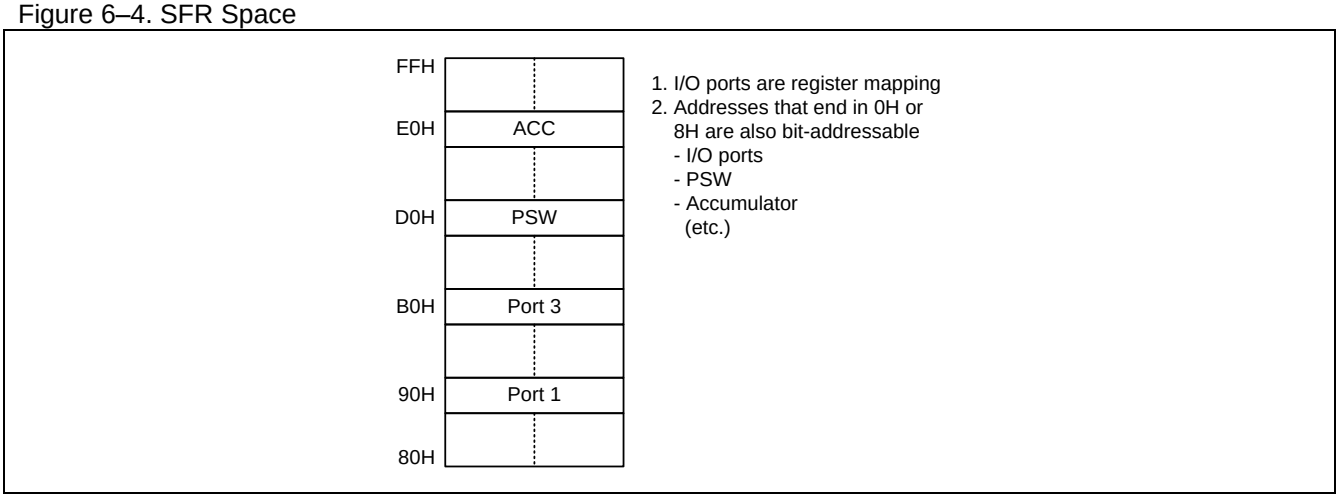


Figure 6–4. SFR Space



## 6.3. Declaration Identifiers in a C51-Compiler

The declaration identifiers in a C51-compiler for the various **MG86FE/L104** memory spaces are as follows:

***data***
128 bytes of internal data memory space (00h~7Fh); accessed via direct or indirect addressing, using instructions other than MOVX and MOVC. All or part of the Stack may be in this area.

***idata***
Indirect data; 256 bytes of internal data memory space (00h~FFh) accessed via indirect addressing using instructions other than MOVX and MOVC. All or part of the Stack may be in this area. This area includes the data area and the 128 bytes immediately above it.

***sfr***
Special Function Registers; CPU registers and peripheral control/status registers, accessible only via direct addressing.

***xdata***
There is no on-chip XRAM or XRAM interface for ***xdata*** access.

***pdata***
There is no on-chip XRAM or XRAM interface for ***pdata*** access.

***code***
4K bytes of program memory space; accessed as part of program execution and via the “MOVC @A+DTPR” instruction.

# 7. Data Pointer Register (DPTR)

There is only one set DPTR in **MG86FE/L104**. And **MG86FE/L104** does not support external memory access and MOVX instruction.

# 8. System Clock

There are four clock sources for the system clock: Internal High-frequency RC Oscillator (IHRCO), external crystal oscillator, Internal Low-frequency RC Oscillator (ILRCO) and External Clock Input. Figure 8–1 shows the structure of the system clock in **MG86FE/L104**.

The **MG86FE/L104** always boots from IHRCO on 24MHz with divided 2 on system clock and reserves crystal pads as P4.0/P4.1 GPIO function. Software can select the one of the four clock sources by application required and switches them on the fly. But software needs to settle the clock source stably before clock switching. If software selects external crystal mode, port pin of P4.0 and P4.1 will be assigned to XTAL2 and XTAL1. And P4.0/P4.1 GPIO function will be inhibited. In external clock input mode (ECKI), the clock source comes from P4.0 input and P4.1 still serves the GPIO function.

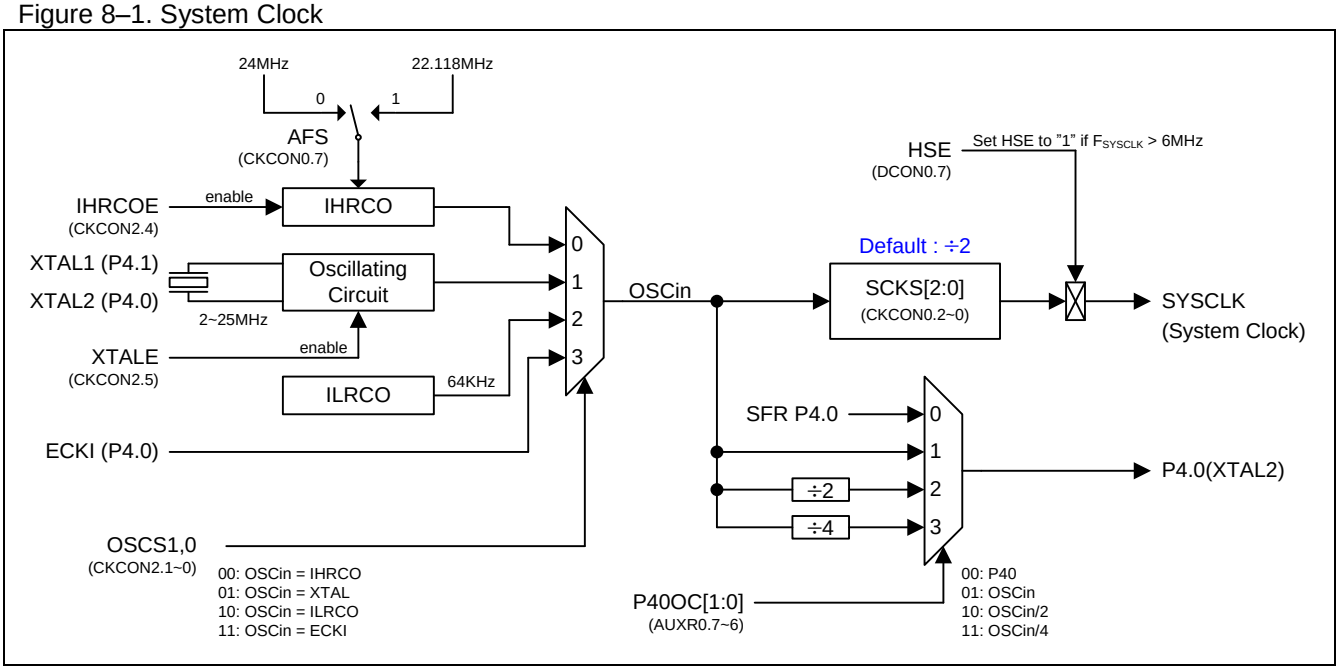
The built-in IHRCO provides two kinds of frequency for software selected. Another frequency is 22.118MHz by software setting AFS on CKCON0.7. Both of 24MHz and 22.118 MHz in IHRCO provide high precision frequency with **±4%** deviation over all operating voltage and temperature. In IHRCO or ILRCO mode, P4.0 can be configured to internal *OSCin* output or *OSCin*/2 and *OSCin*/4 for system application.

The system clock, *SYSCLK*, is obtained from one of these four clock sources through the clock divider, as shown in Figure 8–1. The user can program the divider control bits SCKS2~SCKS0 (in CKCON0 register) to get the desired system clock. The default system clock divider is set to “/2” in **MG86FE/L104** after power on or reset.

## 8.1. Clock Structure

Figure 8–1 presents the principal clock systems in the **MG86FE/L104**. The system clock can be sourced by the external oscillator circuit or either internal oscillator.

Figure 8–1. System Clock



## 8.2. Clock Register

***CKCON0: Clock Control Register 0***

SFR Page = Normal & Page P
SFR Address = 0xC7 RESET = 0xxx-x001

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| AFS | 0 | 0 | 0 | 0 | SCKS2 | SCKS1 | SCKS0 |
| **R/W** | **W** | **W** | **W** | **W** | **R/W** | **R/W** | **R/W** |

Bit 7: AFS, Alternated Frequency Selection.
0: Select IHRCO to output 24MHz.
1: Select IHRCO to output 22.118MHz.

Bit 6~3: Reserved. Software must write “0” on these bits when CKCON0 is written.

Bit 2~0: SCKS2 ~ SCKS0, programmable System Clock Selection. The default value of SCKS[2:0] is set to “001” to select system clock on OSCin/2.

| SCKS[2:0] | System Clock |
|---|---|
| 0 0 0 | OSCin |
| 0 0 1 | OSCin /2 (default setting) |
| 0 1 0 | OSCin /4 |
| 0 1 1 | OSCin /8 |
| 1 0 0 | OSCin /16 |
| 1 0 1 | OSCin /32 |
| 1 1 0 | OSCin /64 |
| 1 1 1 | OSCin /128 |

***CKCON2: Clock Control Register 2***

SFR Page = Page P Only
SFR Address = 0x40 RESET = xx01-xx00

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| 0 | 0 | XTALE | IHRCOE | 0 | 0 | OSCS1 | OSCS0 |
| **W** | **W** | **R/W** | **R/W** | **W** | **W** | **R/W** | **R/W** |

Bit 7~6: Reserved. Software must write “0” on these bits when CKCON2 is written.

Bit 5: XTALE, external Crystal (XTAL) Enable. The default value is set by hardware option on clock source selection.
0: Disable XTAL oscillating circuit. In this case, XTAL2 and XTAL1 behave as Port 4.0 and Port 4.1.
1: Enable XTAL oscillating circuit. If this bit is set by CPU software, it needs **3 ms** to have stable output after XTALE enabled.

Bit 4: IHRCOE, Internal High frequency RC Oscillator Enable.
0: Disable internal high frequency RC oscillator.
1: Enable internal high frequency RC oscillator. If this bit is set by CPU software, it needs **32 us** to have stable output after IHRCOE enabled.

Bit 3~2: Reserved. Software must write “0” on these bits when CKCON2 is written.

Bit 1~0: OSCS[1:0], OSCin source selection. The default selection of OSCin is IHRCO.

| OSCS[1:0] | OSCin source Selection |
|---|---|
| 0 0 | IHRCO |
| 0 1 | XTAL |
| 1 0 | ILRCO |
| 1 1 | ECKI, External Clock Input (P4.0) as OSCin. |

***AUXR0: Auxiliary Register 0***

SFR Page = Normal

SFR Address = 0xA1 RESET = 000x-0000

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| P40OC1 | P40OC0 | P40FD | -- | P1FS1 | P1FS0 | INT1H | INT0H |
| **R/W** | **R/W** | **R/W** | **W** | **R/W** | **R/W** | **R/W** | **R/W** |

Bit 7~6: P4.0 function configured control bit 1 and 0. The two bits only act when internal RC oscillator (IHRCO or ILRCO) is selected for system clock source. In crystal mode, XTAL2 and XTAL1 are the alternated function for P4.0 and P4.1. In external clock input mode, P4.0 is the dedicated clock input pin. In internal oscillator condition, P4.0 provides the following selections for GPIO or clock source generator. When P40OC[1:0] index to non-P4.0 GPIO function, P4.0 will drive the on-chip RC oscillator (IHRCO or ILRCO) output to provide the clock source for other devices.

| P40OC[1:0] | P4.0 function | P4.0 I/O mode |
|---|---|---|
| 00 | P40 | By P4M0.0 |
| 01 | OSCin | By P4M0.0 |
| 10 | OSCin/2 | By P4M0.0 |
| 11 | OSCin/4 | By P4M0.0 |

For clock-out on P4.0 function, it is recommended to set P4M0.0 to “1” which selects P4.0 as push-push output mode.

Bit 5: P40FD, P4.0 Fast Driving.
0: P4.0 output with default driving.
1: P4.0 output with fast driving enabled. If P4.0 is configured to clock output, enable this bit when P4.0 output frequency is more than 12MHz at 5V application or more than 6MHz at 3V application.

***DCON0: Device Control 0***

SFR Page = Page P only

SFR Address = 0x4C RESET = 10xx-xx1x

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| HSE | IAPO | 0 | 0 | 0 | 0 | RSTIO | 0 |
| **W** | **W** | **W** | **W** | **W** | **W** | **W** | **W** |

Bit 7: HSE, High Speed operation Enable.
0: Select MCU running in low speed mode which is slow down internal circuit to reduce power consumption.
1: Enable MCU full speed operation if $F_{SYSCLK}$ > 6MHz.

## 8.3. Clock Sample Code

*(1) Required function: Switch IHRCO from default 24MHz to 22.118MHz*

Assembly Code Example:

```
ORL     CKCON0,#(AFS)                   ; Select IHRCO to output 22.118MHz
```

C Code Example:

```
CKCON0 |= AFS;                          // Select IHRCO to output 22.1184MHz.
```

*(2). Required Function: Switch SYSCLK to OSCin/1 (default is OSCin/2)*

Assembly Code Example:

```
ANL     CKCON0,#(AFS)                   ; Set SCKS[2:0] = 0 to select OSCin/1
```

C Code Example:

```
CKCON0 &= ~(SCKS2 | SCKS1 | SCKS0);     // System clock divider /1
                                        // SCKS[2:0], system clock divider
                                        //     0      | OSCin/1
                                        //     1      | OSCin/2
                                        //     2      | OSCin/4
                                        //     3      | OSCin/8
                                        //     4      | OSCin/16
                                        //     5      | OSCin/32
                                        //     6      | OSCin/64
                                        //     7      | OSCin/128
```

*(3). Required Function: Select XTAL as OSCin source when MCU using IHRCO or ILRCO (default is IHRCO)*

Assembly Code Example:

```
    MOV     IFADRL,#(CKCON2)            ; Index Page-P address to CKCON2
    CALL    _page_p_sfr_read            ; Read CKCON2 data

    ORL     IFD,#( XTALE)               ; Enable XTALE
                                        ;
    CALL    _page_p_sfr_write           ; Write data to CKCON2, SYSCLK must be less than 25MHz

check_XTOR:                             ; Check XTAL oscillating ready
    MOV     A,AUXR1
    JNB     ACC.4,check_XTOR            ; Waiting for XTOR(AUXR1.4) true

    ANL     IFD,#~(OSCS1 | OSCS0)       ; Switch OSCin source to XTAL.
    ORL     IFD,#(OSCS0)
    CALL    _page_p_sfr_write           ; Write data to CKCON2

    ANL     IFD,#~(IHRCOE)              ; Disable IHRCO if MCU is switched from IHRCO
    CALL    _page_p_sfr_write           ; Write data to CKCON2
```

C Code Example:

```
    IFADRL = CKCON2;                    // Index Page-P address to CKCON2
    page_p_sfr_read();                  // Read CKCON2 data

    IFD |= XTALE;                       // Enable XTALE
    page_p_sfr_write ();                // Write data to CKCON2, SYSCLK must be less than 25MHz

    while(AUXR1 & XTOR == 0x00);        // Check XTAL oscillating ready
                                        // Waiting for XTOR(AUXR1.4) true

    IFD &= ~(OSCS1 | OSCS0);            // Switch OSCin source to XTAL.
```

```
IFD |= OSCS0;
page_p_sfr_write ();                    // Write data to CKCON2

IFD &= ~IHRCOE;                         // Disable IHRCO if MCU is switched from IHRCO
page_p_sfr_write();                     // Write data to CKCON2
```

*(4). Required Function: Select ILRCO as OSCin source when MCU using IHRCO, ECKI or XTAL (default is IHRCO)*

Assembly Code Example:

```
MOV     IFADRL,#(CKCON2)            ; Index Page-P address to CKCON2
CALL    _page_p_sfr_read            ; Read CKCON2 data

ANL     IFD,#~(OSCS1 | OSCS0)       ; Switch OSCin source to ILRCO
ORL     IFD,#(OSCS1)
CALL    _page_p_sfr_write           ; Write data to CKCON2

ANL     IFD,#~(XTALE | IHRCOE)      ; Disable XTAL and IHRCO
CALL    _page_p_sfr_write           ; Write data to CKCON2

MOV     IFADRL,#(DCON0)             ; Index Page-P address to DCON0
CALL    _page_p_sfr_read            ; Read DCON0 data

ANL     IFD,#~(HSE)                 ; Disable HSE when SYSCLK ≤ 6MHz for power saving
CALL    _page_p_sfr_write           ; Write data to DCON0
```

C Code Example:

```
IFADRL = CKCON2;                    // Index Page-P address to CKCON2
page_p_sfr_read();                  // Read CKCON2 data.

IFD = ~(OSCS1 | OSCS0);             // Switch OSCin source to ILRCO
IFD |= OSCS1;
page_p_sfr_write();                 // Write data to CKCON2

IFD &= ~(XTALE | IHRCOE);           // Disable XTAL and IHRCO
page_p_sfr_write();                 // Write data to CKCON2

IFADRL = DCON0;                     // Index Page-P address to DCON0
page_p_sfr_read();                  // Read DCON0 data

IFD &= ~HSE;                        // Disable HSE when SYSCLK ≤ 6MHz for power saving
page_p_sfr_write();                 // Write data to DCON0
```

*(5). Required Function: Select ECKI as OSCin source when MCU using IHRCO or ILRCO (default is IHRCO)*

Assembly Code Example:

```
MOV     IFADRL,#(CKCON2)            ; Index Page-P address to CKCON2
CALL    _page_p_sfr_read            ; Read CKCON2 data

ORL     IFD,#(OSCS1 | OSCS0)        ; Switch OSCin source to ECKI
CALL    _page_p_sfr_write           ; Write data to CKCON2, SYSCLK must be less than 25MHz

ANL     IFD,#~(XTALE | IHRCOE)      ; Disable IHRCO & XTAL
CALL    _page_p_sfr_write           ; Write data to CKON2
```

C Code Example:

```
IFADRL = CKCON2;                    // Index Page-P address to CKCON2
page_p_sfr_read();                  // Read CKCON2 data.

IFD |= OSCS1 | OSCS0;               // Switch OSCin source to ECKI
page_p_sfr_write ();                // Write data to CKCON2, SYSCLK must be less than 25MHz
```

```
IFD &= ~(XTALE | IHRCOE);                  //Disable IHRCO and XTAL
page_p_sfr_write ();                       // Write data to CKCON2
```

*(6). Required Function: Select IHRCO as OSCin source when MCU using ILRCO, ECKI or XTAL*

Assembly Code Example:

```
MOV      IFADRL,#(CKCON2)          ; Index Page-P address to CKCON2
CALL     _page_p_sfr_read          ; Read CKCON2 data

ORL      IFD,#(IHRCOE)             ; Enable IHRCO
CALL     _page_p_sfr_write         ; Write data to CKCON2

Delay_32us

ANL      IFD,#~(OSCS1 | OSCS0)     ; Switch OSCin source to IHRCO
CALL     _page_p_sfr_write         ; Write data to CKCON2

ANL      IFD,#~(XTALE)             ; Disable XTAL
CALL     _page_p_sfr_write         ; Write data to CKCON2
```

C Code Example:

```
IFADRL = CKCON2;                   // Index Page-P address to CKCON2
page_p_sfr_read();                 // Read CKCON2 data.

IFD |= IHRCOE;                     // Enable IHRCO
page_p_sfr_write();                // Write data to CKCON2

Delay 32us

IFD &= ~(OSCS1 | OSCS0);           // Switch OSCin source to IHRCO
page_p_sfr_write();                // Write data to CKCON2

IFD &= ~ XTALE;                    // Disable XTAL
page_p_sfr_write();                // Write data to CKCON2
```

*(7). Required Function: Output IHRCO frequency on P4.0*

Assembly Code Example:

```
MOV      P4M0,#P4M00               ; Set P4.0 to push-pull output mode
ANL      AUXR0,#~(P40OC1|P40OC0)    ; Switch P4.0 to GPIO function
ORL      AUXR0,#(P40OC0|P4FD)      ; P4.0 = IHRCO Frequency + Pin fast driving
                                   ; P40OC[1:0]   |   P4.0
                                   ; 00           |   GPIO
                                   ; 01           |   IHRCO/1
                                   ; 10           |   IHRCO/2
                                   ; 11           |   IHRCO/4
```

C Code Example:

```
P4M0 |= P4M00;                     // P4.0 select push-pull output mode.
AUXR0 &= ~(P40OC0 | P40OC1);        // Switch P4.0 to GPIO function
AUXR0 |= (P40OC0 | P4FD);           // P4.0 output IHRCO/1
//   AUXR0 = P40OC1|P4FD;           // P4.0 output IHROC/2
//   AUXR0 = P40OC1|P40OC0|P4FD;    // P4.0 output IHRCO/4
```

# 9. Watch Dog Timer (WDT)

## 9.1. WDT Structure

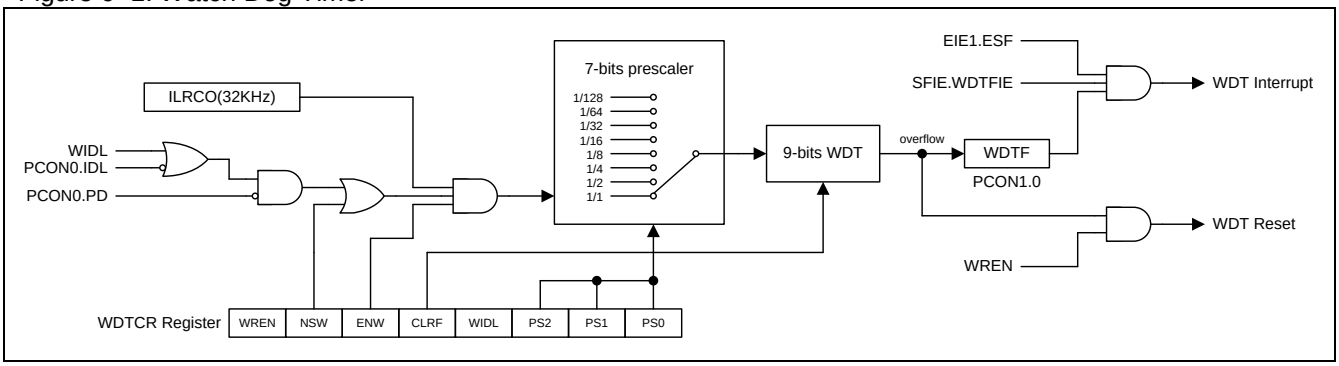
The Watch-dog Timer (WDT) is intended as a recovery method in situations where the CPU may be subjected to software upset. The WDT consists of a 9-bit free-running counter, a 7-bit prescaler and a control register (WDTCR). Figure 9–1 shows the WDT structure in **MG86FE/L104**.

When WDT is enabled, it derives its time base from the 64KHz ILRCO. The WDT overflow will set the WDTF on PCON1.0 which can be configured to generate an interrupt by enabled WDTFIE (SFIE.0) and enabled ESF (EIE1.3). The overflow can also trigger a system reset when WREN (WDTCR.7) is set. To prevent WDT overflow, software needs to clear it by writing "1" to the CLRW bit (WDTCR.4) before WDT overflows.
*Note: the WDTFIE function is under verifying.*

Once the WDT is enabled by setting ENW bit, there is no way to disable it except through power-on reset or page-p SFR over-write on ENW, which will clear the ENW bit. The WDTCR register will keep the previous programmed value unchanged after hardware (RST-pin) reset, software reset and WDT reset.

WREN, NSW and ENW are implemented to one-time-enabled function, only writing "1" valid in general SFR page. Page-P SFR Access on WDTCR can disable WREN, NSW and ENW, writing "0" on WDTCR.7~5. Please refer Section "9.3 WDT Register" and Section "18 Page P SFR Access" for more detail information.

Figure 9–1. Watch Dog Timer




## 9.2. WDT During Idle and Power Down

In the Idle mode, the WIDL bit (WDTCR.3) determines whether WDT counts or not. Set this bit to let WDT keep counting in the Idle mode. If the hardware option WDTRCO is enabled, the WDT always keeps counting regardless of WIDL bit.

In the Power down mode, the ILRCO won't stop if the NSW (WDTCR.6) is enabled. That lets WDT keep counting even in Power down mode (Watch Mode). After WDT overflows, it will wake up the CPU from interrupt or reset by software configured.

## 9.3. WDT Register

***WDTCR: Watch-Dog-Timer Control Register***

SFR Page = Normal & Page P

SFR Address = 0xE1 POR = 0000-0111 (xxx0_xxxx by Hardware Option)

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| WREN | NSW | ENW | CLRW | WIDL | PS2 | PS1 | PS0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

Bit 7: WREN, WDT Reset Enable. The initial value can be changed by hardware option, WRENO.
0: The overflow of WDT does not set the WDT reset. The WDT overflow flag, WDTF, may be polled by software or trigger an interrupt.
1: The overflow of WDT will cause a system reset. Once WREN has been set, it can not be cleared by software in page 0~F. **In page P, software can modify it to "0" or "1".**

Bit 6: NSW. Non-Stopped WDT. The initial value can be changed by hardware option, NSWDT.
0: WDT stop counting while the MCU is in power-down mode.
1: WDT always keeps counting while the MCU is in power-down mode (Watch Mode) or idle mode. Once NSW has been set, it can not be cleared by software in general page. **In page P, software can modify it to "0" or "1".**

Bit 5: ENW. Enable WDT.
0: Disable WDT running. This bit is only cleared by POR.
1: Enable WDT while it is set. Once ENW has been set, it can not be cleared by software in general page. **In Page P, software can modify it as "0" or "1".**

Bit 4: CLRW. Clear WDT counter.
0: Hardware will automatically clear this bit.
1: Clear WDT to recount while it is set.

Bit 3: WIDL. WDT idle control.
0: WDT stops counting while the MCU is in idle mode.
1: WDT keeps counting while the MCU is in idle mode.

Bit 2~0: PS2 ~ PS0, select prescaler output for WDT time base input.

| PS[2:0] | Prescaler Value | WDT Period |
|---|---|---|
| 0 0 0 | 1 | 15 ms |
| 0 0 1 | 2 | 31 ms |
| 0 1 0 | 4 | 62 ms |
| 0 1 1 | 8 | 124 ms |
| 1 0 0 | 16 | 248 ms |
| 1 0 1 | 32 | 496 ms |
| 1 1 0 | 64 | 992 ms |
| 1 1 1 | 128 | 1.984 S |

***PCON1: Power Control Register 1***

SFR Page = Normal & Page P

SFR Address = 0x97 POR = 0010-0x00

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| SWRF | EXRF | -- | -- | KBIF | -- | BOF0 | WDTF |
| R/W | R/W | W | W | R/W | W | R/W | R/W |

Bit 0: WDTF, WDT overflow flag.
0: This bit must be cleared by software writing **"1"** on it. Software writing ":0" is no operation.
1: This bit is only set by hardware when WDT overflows. Writing **"1"** on this bit will clear WDTF.

## 9.4. WDT Hardware Option

In addition to being initialized by software, the WDTCR register can also be automatically initialized at power-up by the hardware options WRENO, NSWDT, HWENW, HWWIDL and HWPS[2:0], which should be programmed by a universal Writer or Programmer, as described below.

If HWENW is programmed to “enabled”, then hardware will automatically do the following initialization for the WDTCR register at power-up: (1) set ENW bit, (2) load WRENO into WREN bit, (3) load NSWDT into NSW bit, (4) load HWWIDL into WIDL bit, and (5) load HWPS[2:0] into PS[2:0] bits.

If both of HWENW and WDSFWP are programmed to “enabled”, hardware still initializes the WDTCR register content by WDT hardware option at power-up. Then, any CPU writing on WDTCR bits will be inhibited except writing “1” on WDTCR.4 (CLRW), clear WDT, even though access through Page-P SFR mechanism.

**WRENO:**
☑: Enabled. Set WDTCR.WREN to enable a system reset function by WDTF.
☐: Disabled. Clear WDTCR.WREN to disable the system reset function by WDTF.

**NSWDT:** Non-Stopped WDT
☑: Enabled. Set WDTCR.NSW to enable the WDT running in power down mode (watch mode).
☐: Disabled. Clear WDTCR.NSW to disable the WDT running in power down mode (disable Watch mode).

**HWENW:** Hardware loaded for “ENW” of WDTCR.
☑: Enabled. Enable WDT and load the content of WRENO, NSWDT, HWWIDL and HWPS2~0 to WDTCR after power-on.
☐: Disabled. WDT is not enabled automatically after power-on.

**HWWIDL, HWPS2, HWPS1, HWPS0:**
When HWENW is enabled, the content on these four fused bits will be loaded to WDTCR SFR after power-on.

**WDSFWP:**
☑: Enabled. The WDT SFRs, WREN, NSW, WIDL, PS2, PS1 and PS0 in WDTCR, will be write-protected.
☐: Disabled. The WDT SFRs, WREN, NSW, WIDL, PS2, PS1 and PS0 in WDTCR, are free for writing of software.

## 9.5. WDT Sample Code

*(1) Required function: Enable WDT and select WDT period to 248ms*

Assembly Code Example:

```
ANL     PCON1,#(WDTF)                   ; Clear WDTF flag (write “1”)
MOV     WDTCR,#(ENW | CLRW | PS2)       ; Enable WDT counter and set WDT period to 248ms
```

C Code Example:

```
PCON1 &= WDTF;                          // Clear WDT flag (write “1”)
WDTCR = (ENW | CLRW | PS2);             // Enable WDT counter and set WDT period to 248ms
                                        // PS[2:0]   | WDT period selection
                                        //      0    | 15ms
                                        //      1    | 31ms
                                        //      2    | 62ms
                                        //      3    | 124ms
                                        //      4    | 248ms
                                        //      5    | 496ms
                                        //      6    | 992ms
                                        //      7    | 1.984s
```

*(2) Required function: How to Disable WDT*

Assembly Code Example:

```
MOV     IFD,WDTCR                       ; Read WDTCR data
ANL     IFD,#~(ENW)                     ; Clear ENW to disable WDT

MOV     IFADRL,#(WDTCR_P)               ; Index Page-P address to WDTCR_P
CALL    _page_p_sfr_write               ; Write data to WDTCR
```

C Code Example:

```
IFD = WDTCR;                            // Read WDTCR data
IFD &= ~ENW;                            // Clear ENW to disable WDT

IFADRL = WDTCR_P;                       // Index Page-P address to WDTCR_P
page_p_sfr_write();                     // Write data to WDTCR
```

*(3). Required Function: Enable WDT reset function and select WDT period to 62ms*

Assembly Code Example:

```
ANL     PCON1,#(WDTF)                   ; Clear WDTF flag (write “1”)
MOV     WDTCR,#(WREN | CLRW | PS1)      ; Enable WDT reset function and set WDT period to 62ms

ORL     WDTCR,#(ENW)                    ; Enable WDT counter, WDT running
```

C Code Example:

```
PCON1 &= WDTF;                          // Clear WDTF flag (write “1”)
WDTCR = WREN | CLRW | PS1;              // Enable WDT reset function and set WDT period to 62ms

WDTCR |= ENW;                           // Enable WDT counter, WDT running.
```

*(4). Required Function: Enable protected write for WDTCR*

Assembly Code Example:

```
ANL     PCON1,#(WDTF)                ; Clear WDTF flag (write “1”)
MOV     WDTCR,#(ENW | CLRW | PS2)    ; Enable WDT counter and set WDT period to 248ms

MOV     IFADRL,#(SPCON0)             ; Index Page-P address to SPCON0
CALL    _page_p_sfr_read             ; Read SPCON0 data

ORL     IFD,#(WRCTL)                 ; Enable protected write for WDTCR
CALL    _page_p_sfr_write            ; Write data to SPCON0

MOV     IFD,WDTCR                    ; Read WDTCR data
ORL     IFD,#(CLRW)                  ; Enable CLRW

MOV     IFADRL,#(WDTCR_P)            ; Index Page-P address to WDTCR_P
CALL    _page_p_sfr_write            ; Write data to WDTCR to clear WDT counter
```

C Code Example:

```
PCON1 &= WDTF;                       // Clear WDTF flag (write “1”)
WDTCR = ENW | CLRW | PS2;            // Enable WDT counter and set WDT period to 248ms

IFADRL = SPCON0;                     // Index Page-P address to SPCON0
page_p_sfr_read();                   // Read SPCON0 data

IFD |= WRCTL;                        // Enable protected write for WDTCR
page_p_sfr_write();                  // Write data to SPCON0

IFD = WDTCR;                         // Read WDTCR data
IFD |= CLRW;                         // Enable CLRW

IFADRL = WDTCR_P;                    // Index Page-P address to WDTCR_P
page_p_sfr_write();                  // Write data to WDTCR to clear WDT counter
```

# 10. System Reset

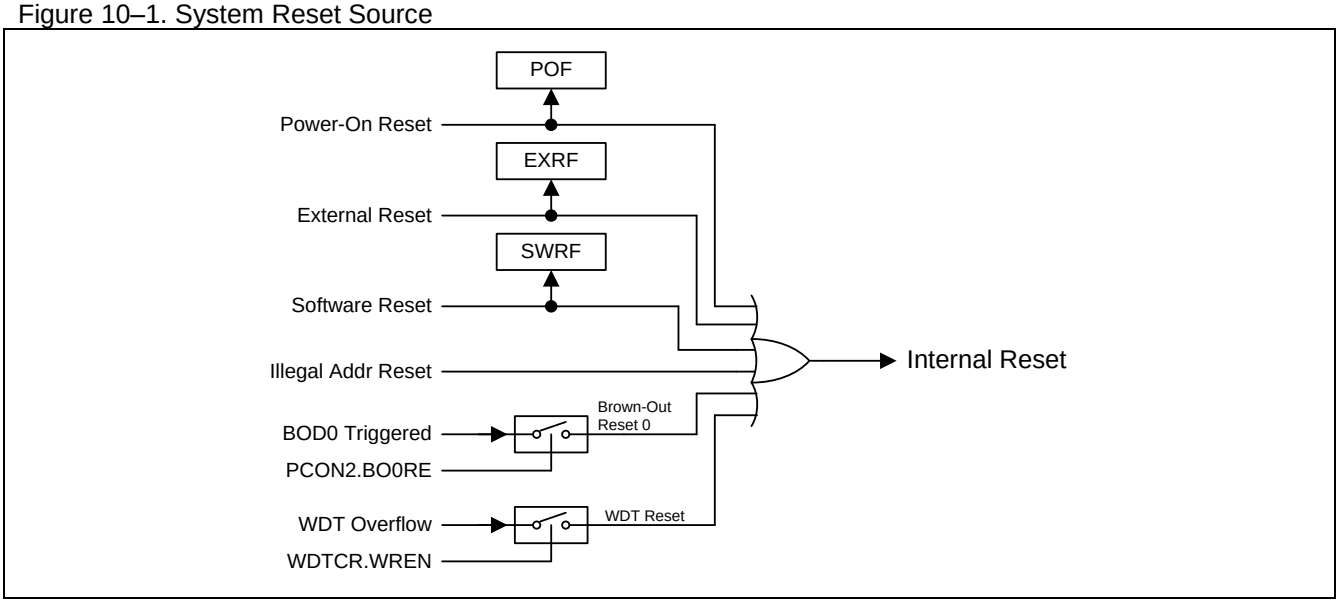
During reset, all I/O Registers are set to their initial values, and the program starts execution from the Reset Vector, 0000H, or ISP start address by OR setting. The **MG86FE/L104** has six sources of reset: power-on reset, external reset, software reset, illegal address reset, WDT reset and brown-out reset. Figure 10–1 shows the system reset source in **MG86FE/L104**.

The following sections describe the reset happened source and corresponding control registers and indicating flags.

## 10.1. Reset Source

Figure 10–1 presents the reset systems in the **MG86FE/L104** and all of its reset sources.

Figure 10–1. System Reset Source




## 10.2. Power-On Reset

Power-on reset (POR) is used to internally reset the CPU during power-up. The CPU will keep in reset state and will not start to work until the VDD power rises above the voltage of Power-On Reset. And, the reset state is activated again whenever the VDD power falls below the POR voltage. During a power cycle, VDD must fall below the POR voltage before power is reapplied in order to ensure a power-on reset

***PCON0: Power Control Register 0***

SFR Page = Normal & Page P

SFR Address = 0x87 POR = 0001-0000, RESET = 000X-0000

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| SMOD1 | SMOD0 | GF | **POF** | GF1 | GF0 | PD | IDL |
| **R/W** | **R/W** | **R/W** | **R/W** | **R/W** | **R/W** | **R/W** | **R/W** |

Bit 4: POF. Power-On Flag.
0: The flag must be cleared by software to recognize next reset type.
1: Set by hardware when VDD rises from 0 to its nominal voltage. POF can also be set by software.

The Power-on Flag, POF, is set to “1” by hardware during power up or when VDD power drops below the POR voltage. It can be clear by firmware and is not affected by any warm reset such as external reset, Brown-Out reset, software reset (ISPCR.5) and WDT reset. It helps users to check if the running of the CPU begins from power up or not. Note that the POF must be cleared by firmware.

## 10.3. External Reset

A reset is accomplished by holding the RESET pin HIGH for at least 24 oscillator periods while the oscillator is running. To ensure a reliable power-up reset, the hardware reset from RST pin is necessary.

***PCON1: Power Control Register 1***

SFR Page = Normal & Page P
SFR Address = 0x97 POR = 0010-0x00

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| SWRF | **EXRF** | -- | -- | KBIF | -- | BOF0 | WDTF |
| **R/W** | **R/W** | **W** | **W** | **R/W** | **W** | **R/W** | **R/W** |

Bit 6: EXRF, External Reset Flag.
0: This bit must be cleared by software writing **"1"** on it. Software writing ":0" is no operation.
1: This bit is only set by hardware if an External Reset occurs. Writing **"1"** on this bit will clear EXRF.

## 10.4. Software Reset

Software can trigger the CPU to restart by software reset, writing "1" on SWRST (ISPCR.5), and set the SWRF flag (PCON1.7). SWBS decides the CPU is boot from ISP or AP region after the reset action

***ISPCR: ISP Control Register***

SFR Page = Normal
SFR Address = 0xE5 RESET = 0000-xxxx

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| ISPEN | SWBS | **SWRST** | CFAIL | - | -- | -- | -- |
| **R/W** | **R/W** | **R/W** | **R/W** | **W** | **W** | **W** | **W** |

Bit 6: SWBS, software boot selection control.
0: Boot from AP-memory after reset.
1: Boot from ISP memory after reset.

Bit 5: SWRST, software reset trigger control.
0: No operation
1: Generate software system reset. It will be cleared by hardware automatically.

***PCON1: Power Control Register 1***

SFR Page = Normal & Page P
SFR Address = 0x97 POR = 0010-0x00

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| **SWRF** | EXRF | -- | -- | KBIF | -- | BOF0 | WDTF |
| **R/W** | **R/W** | **W** | **W** | **R/W** | **W** | **R/W** | **R/W** |

Bit 7: SWRF, Software Reset Flag.
0: This bit must be cleared by software writing "1" on it. Software writing "0" is no operation.
1: This bit is only set by hardware if a Software Reset occurs. Writing "1" on this bit will clear SWRF.

## 10.5. Illegal Address Reset

In **MG86FE/L104**, if software program runs to illegal address such as over program ROM limitation, it triggers a RESET to CPU.

## 10.6. Brown-Out Reset

In **MG86FE/L104**, there is a Brown-Out Detectors (BOD0) to monitor VDD power. BOD0 services the fixed detection level at VDD=2.4V/4.2V. If VDD power drops below BOD0 monitor level. Associated flag, BOF0, is set when BOD0 meets the detection level. If BO0RE (PCON2.1) is enabled, the BOD0 will trigger a system reset and a set BOF0 indicates a BOD0 Reset occurred.

***PCON1: Power Control Register 1***

SFR Page = Normal & Page P

SFR Address = 0x97 POR = 0010-0x00

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| SWRF | EXRF | -- | -- | KBIF | -- | **BOF0** | WDTF |
| **R/W** | **R/W** | **W** | **W** | **R/W** | **W** | **R/W** | **R/W** |

Bit 1: BOF0, BOF0 (Reset) Flag.

0: This bit must be cleared by software writing **"1"** on it. Software writing ":0" is no operation.

1: This bit is only set by hardware when VDD meets BOD0 monitored level. Writing **"1"** on this bit will clear BOF0. If BO0RE (PCON2.1) is enabled, BOF0 indicates a BOD0 Reset occurred.

## 10.7. WDT Reset

When WDT is enabled to start the counter, WDTF will be set by WDT overflow. If WREN (WDTCR.7) is enabled, the WDT overflow will trigger a system reset that causes CPU to restart. Software can read the WDTF to recognize the WDT reset occurred.

***PCON1: Power Control Register 1***

SFR Page = Normal & Page P

SFR Address = 0x97 POR = 0010-0x00

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| SWRF | EXRF | -- | -- | KBIF | -- | BOF0 | **WDTF** |
| **R/W** | **R/W** | **W** | **W** | **R/W** | **W** | **R/W** | **R/W** |

Bit 0: WDTF, WDT Overflow/Reset Flag.

0: This bit must be cleared by software writing **"1"** on it. Software writing ":0" is no operation.

1: This bit is only set by hardware when WDT overflows. Writing **"1"** on this bit will clear WDTF. If WREN (WDTCR.7) is set, WDTF indicates a WDT Reset occurred.

## 10.8. Reset Sample Code

*(1) Required function: Trigger a software reset*

Assembly Code Example:

```
ORL     ISPCR,#SWRST                ; Trigger Software Reset
```

C Code Example:

```
ISPCR |= SWRST;                     // Trigger Software Reset
```

*(2). Required Function: Enable BOD0 reset*

Assembly Code Example:

```
MOV     IFADRL,#PCON2               ; Index Page-P address to PCON2
CALL    _page_p_sfr_read            ; Read PCON2 data

ORL     IFD,#BO0RE                  ; Enable BOD0 reset function
CALL    _page_p_sfr_write           ; Write data to PCON2
```

C Code Example:

```
IFADRL = PCON2;                     // Index Page-P address to PCON2
page_p_sfr_read();                  // Read PCON2 data

IFD |= BO0RE;                       // Enable BOD0 reset function
page_p_sfr_write();                 // Write data to PCON2
```

# 11. Power Management

The **MG86FE/L104** supports one power monitor module, Brown-Out Detector (BOD0), and 6 power-reducing modes: Idle mode, Power-down mode, Slow mode, Sub-Clock mode, Watch mode and Monitor mode.

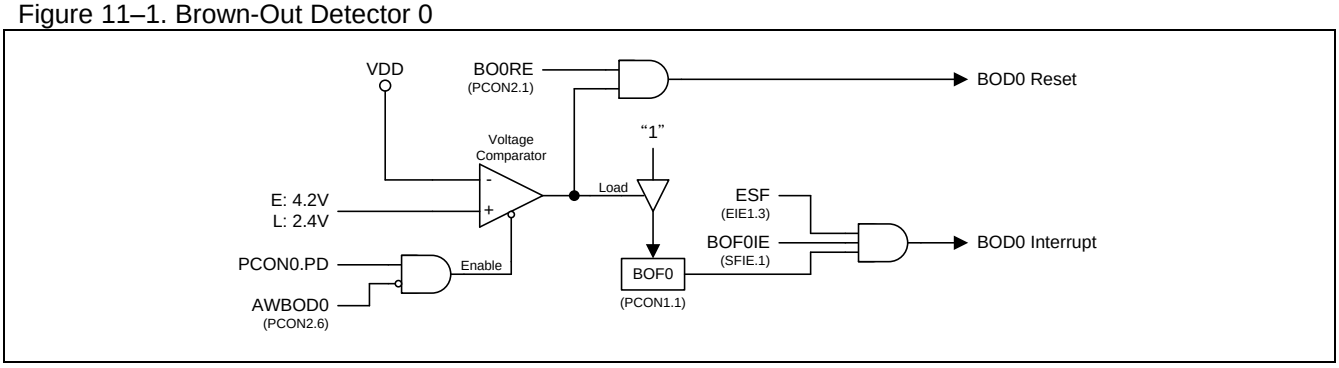
BOD0 reports the chip power status on the flag, BOF0, which provides the capability to interrupt CPU or to reset CPU by software configured. The six power-reducing modes provide the different power-saving scheme for chip application. These modes are accessed through the CKCON0, CKCON2, PCON0, PCON1, PCON2 and WDTCR register.

## 11.1. Brown-Out Detector

In **MG86FE/L104**, there is a Brown-Out Detectors (BOD0) to monitor VDD power. Figure 11–1 shows the functional diagram of BOD0. BOD0 services the fixed detection level at VDD=4.2V for **5V** part and 2.4V on **3.3V** part. Associated flag, BOF0 (PCON1.1), is set when BOD0 meets the detection level. If both of ESF (EIE1.3) and BOF0IE (SFIE.1) are enabled, a set BOF0 will generate a system flag interrupt. It can interrupt CPU either CPU in normal mode or idle mode. The interrupt also wakes up CPU in power down mode if AWBOD0 (PCON2.6) is enabled.

If BO0RE (PCON2.1) is enabled, the BOD0 event will trigger a system reset and set BOF0 to indicate a BOD0 Reset occurred. The BOD0 reset restart the CPU either CPU in normal mode or idle mode. The reset also restart CPU in power down mode if AWBOD0 (PCON2.6) is enabled in BOD0 reset operation.

Figure 11–1. Brown-Out Detector 0




## 11.2. Power Saving Mode

### 11.2.1. Slow Mode

The alternative to save the operating power is to slow down the MCU's operating speed by programming SCKS2~SCKS0 bits (in CKCON0 register, see Section "8 System Clock") to a non-0/0/0 value. The user should examine which program segments are suitable for lower operating speed. In principle, the lower operating speed should not affect the system's normal function. Then, restore its normal speed in the other program segments.

### 11.2.2. Sub-Clock Mode

The alternative to slow down the MCU's operating speed by programming OSCS1~0 can select the ILRCO for system clock. The 64KHz ILRCO provides the MCU to operates in an ultra low speed and low power operation. Additional programming SCKS2~SCKS0 bits (in CKCON0 register, see Section "8 System Clock"), the user could put the MCU speed down to 500Hz slowest.

### 11.2.3. Watch Mode

If Watch-Dog-Timer is enabled and NSW is set, Watch-Dog-Timer will keep running in power down mode, which named Watch Mode in **MG86FE/L104**. When WDT overflows, set WDTF and wakeup CPU from interrupt or system reset by software configured. The maximum wakeup period is about 2 seconds that is defined by WDT pre-scaler. Please refer Section "9 Watch Dog Timer (WDT)" and Section "13 Interrupt" for more detail information.

### 11.2.4. Monitor Mode (L-Series Only)

If AWBOD1 (PCON3.3) is set, BOD1 will keep VDD monitor in power down mode. It is the Monitor Mode in **MG86FE/L104**. When BOD1 meets the detection level, set BOF1 and wakeup CPU from interrupt or system reset by software configured. Please refer Section "11.1 Brown-Out Detector" and Section "13 Interrupt" for more detail information. This function is only valid in L-series device.

### 11.2.5. Idle Mode

Setting the IDL bit in PCON enters idle mode. Idle mode halts the internal CPU clock. The CPU state is preserved in its entirety, including the RAM, stack pointer, program counter, program status word, and accumulator. The Port pins hold the logical states they had at the time that Idle was activated. Idle mode leaves the peripherals running in order to allow them to wake up the CPU when an interrupt is generated. Timer 0, Timer 1, UART, KBI and the BOD0 will continue to function during Idle mode. The WDT is conditional enabled during Idle mode to wake up CPU. Any enabled interrupt source or reset may terminate Idle mode. When exiting Idle mode with an interrupt, the interrupt will immediately be serviced, and following RETI, the next instruction to be executed will be the one following the instruction that put the device into Idle.

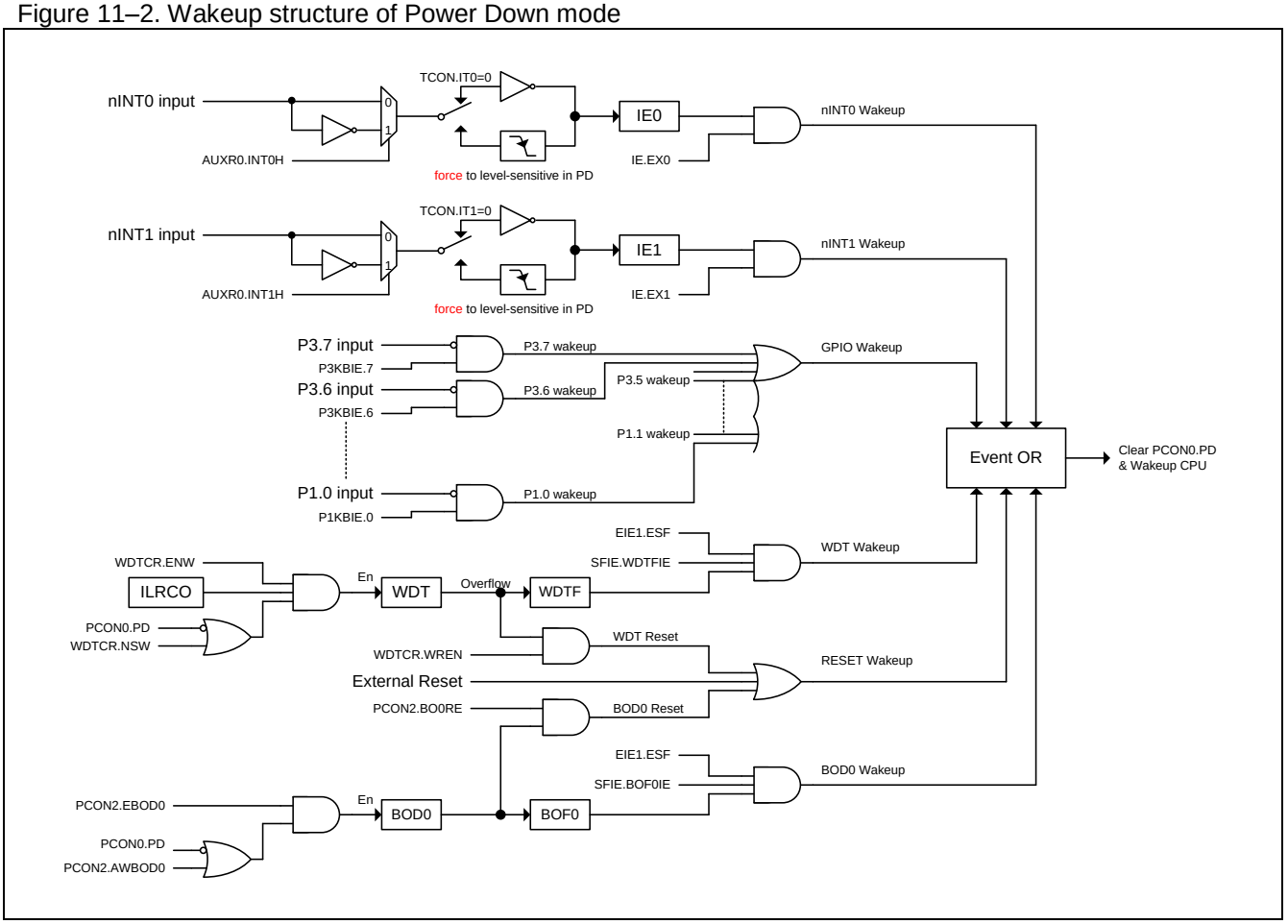
### 11.2.6. Power-Down Mode

Setting the PD bit in PCON enters Power-down mode. Power-down mode stops the oscillator and powers down the Flash memory in order to minimize power consumption. Only the power-on circuitry will continue to draw power during Power-down. During Power-down the power supply voltage may be reduced to the RAM keep-alive voltage. The RAM contents will be retained; however, the SFR contents are not guaranteed once VDD has been reduced. Power-down may be exited by external reset, power-on reset, enabled external interrupts, enabled KBI GPIOs, enabled BOD0 or enabled Non-Stop WDT.

The user should not attempt to enter (or re-enter) the power-down mode for a minimum of 4 µs until after one of the following conditions has occurred: Start of code execution (after any type of reset), or Exit from power-down mode. To ensure minimum power consumption in power down mode, software must confirm all I/O not in floating state, including the port I/Os un-appearance on package pins. For example, P1.7~P1.0 and P4.1~P4.0 are un-appearance in MG86FE/L104AS8 (SOP8) package pins. Software may configure P1/P4 corresponding bit SFR to "0" (output low) to avoid pin floating in power-down mode.

Figure 11–2 shows the wakeup mechanism of power-down mode in **MG86FE/L104**.

Figure 11–2. Wakeup structure of Power Down mode




## 11.2.7. Interrupt Recovery from Power-down

Two external interrupts may be configured to terminate Power-down mode. External interrupts nINT0 (P3.2), nINT1 (P3.3) may be used to exit Power-down. To wake up by external interrupt nINT0, nINT1, the interrupt must be enabled and configured for level-sensitive operation. If the enabled external interrupts are configured to edge-sensitive operation (Falling or Rising), they will be forced to level-sensitive operation (Low level or High level) by hardware in power-down mode.

When terminating Power-down by an interrupt, the wake up period is internally timed. At the falling edge on the interrupt pin, Power-down is exited, the oscillator is restarted, and an internal timer begins counting. The internal clock will not be allowed to propagate and the CPU will not resume execution until after the timer has reached internal counter full. After the timeout period, the interrupt service routine will begin. To prevent the interrupt from re-triggering, the ISR should disable the interrupt before returning. The interrupt pin should be held low until the device has timed out and begun executing.

## 11.2.8. Reset Recovery from Power-down

If P3.6 is configured for RST input pin, wakeup from Power-down through an external reset is similar to the interrupt. At the rising edge of RST, Power-down is exited, the oscillator is restarted, and an internal timer begins counting. The internal clock will not be allowed to propagate to the CPU until after the timer has reached internal counter full. The RST pin must be held high for longer than the timeout period to ensure that the device is reset properly. The device will begin executing once RST is brought low.

It should be noted that when idle is terminated by a hardware reset, the device normally resumes program execution, from where it left off, up to two machine cycles before the internal reset algorithm takes control. On-chip hardware inhibits access to internal RAM in this event, but access to the port pins is not inhibited. To

eliminate the possibility of an unexpected write to a port pin when Idle is terminated by reset, the instruction following the one that invokes Idle should not be one that writes to a port pin or to external memory.

### 11.2.9. KBI Recovery from Power-down

The GPIOs of **MG86FE/L104**, P1.7 ~ P1.0, P3.7~P3.4, P41, P40, P31 and P30 have wakeup CPU capability that are enabled by individual control bit in P1KBIE and P3KBIE. P3.2/nINT0 and P3.3/nINT1 have the wakeup capability from the interrupt function.

Wakeup from Power-down through an enabled KBI GPIO is similar to the interrupt. At the low-level of enabled KBI GPIO, Power-down is exited, the oscillator is restarted, and an internal timer begins counting. The internal clock will not be allowed to propagate to the CPU until after the timer has reached internal counter full. After the timeout period, CPU will meet a KBI interrupt and execute the interrupt service routine. Please refer Section “16 Keypad Interrupt (KBI)” for more detail information.

## 11.3. Power Control Register

***PCON0: Power Control Register 0***

SFR Page = Normal & Page P
SFR Address = 0x87 POR = 0001-0000, RESET = 000x-0000

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| SMOD1 | SMOD0 | -- | POF | GF1 | GF0 | PD | IDL |
| **R/W** | **R/W** | **R/W** | **R/W** | **R/W** | **R/W** | **R/W** | **R/W** |

Bit 4: POF, Power-On Flag.
0: This bit must be cleared by software writing one to it.
1: This bit is set by hardware if a Power-On Reset occurs.

Bit 1: PD, Power-Down control bit.
0: This bit could be cleared by CPU or any exited power-down event.
1: Setting this bit activates power down operation.

Bit 0: IDL, Idle mode control bit.
0: This bit could be cleared by CPU or any exited Idle mode event.
1: Setting this bit activates idle mode operation.

***PCON1: Power Control Register 1***

SFR Page = Normal & Page P
SFR Address = 0x97 POR = 0010-0x00

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| SWRF | EXRF | -- | -- | **KBIF** | -- | BOF0 | WDTF |
| **R/W** | **R/W** | **W** | **W** | **R/W** | **W** | **R/W** | **R/W** |

Bit 7: SWRF, Software Reset Flag.
0: This bit must be cleared by software writing **"1"** to it.
1: This bit is set by hardware if a Software Reset occurs.

Bit 6: EXRF, External Reset Flag.
0: This bit must be cleared by software writing **"1"** to it.
1: This bit is set by hardware if an External Reset occurs.

Bit 5~4: Reserved. Software must write "0" on these bits when PCON1 is written.

Bit 3: KBIF, KBI Flag.
0: This bit must be cleared by software writing **"1"** to it.
1: This bit is set by hardware if a KBI input event occurs.

Bit 2: Reserved. Software must write "0" on this bit when PCON1 is written.

Bit 1: BOF0, Brown-Out Detection flag 0.
0: This bit must be cleared by software writing **"1"** to it.
1: This bit is set by hardware if the operating voltage matches the detection level of Brown-Out Detector 0 (E: 4.2V, L: 2.4V).

Bit 0: WDTF, WDT overflow flag.
0: This bit must be cleared by software writing **"1"** to it.
1: This bit is set by hardware if a WDT overflow occurs.

***PCON2: Power Control Register 2***

SFR Page = **P Only**

SFR Address = 0x44 POR = x0xx-xx01

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| -- | **AWBOD0** | -- | -- | -- | -- | BO0RE | **1** |
| **W** | **W** | **W** | **W** | **W** | **W** | **W** | **W** |

Bit 7: Reserved. Software must write “0” on this bit when PCON2 is written.

Bit 2: AWBOD0, Awaked BOD0 in PD mode.
0: BOD0 is disabled in power-down mode.
1: BOD0 keeps operation in power-down mode.

Bit 5~2: Reserved. Software must write “0” on these bits when PCON2 is written.

Bit 1: BO0RE, BOD0 Reset Enabled. Its initial is loaded from OR1.BO0REO inverted.
0: Disable BOD0 to trigger a system reset when BOF0 is set.
1: Enable BOD0 to trigger a system reset when BOF0 is set by VDD meets 4.2V(E) or 2.4V(L).

Bit 7: Reserved for test. Software must write “**1**” on this bit when PCON2 is written.

***P1KBIE: Port 1 KBI Enable Control Register***

SFR Page = Normal

SFR Address = 0xD7 RESET = 0000-0000

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| P17KBIE | P16KBIE | P15KBIE | P14KBIE | P13KBIE | P12KBIE | P11KBIE | P10KBIE |
| **R/W** | **R/W** | **R/W** | **R/W** | **R/W** | **R/W** | **R/W** | **R/W** |

Bit 7~0: Keypad Input function enable bit for each Port 1 pins.
0: Disable associated port pin for keypad input function.
1: Enable associated port pin for keypad input function on low level.

***P3KBIE: Port 3 KBI Enable Control Register***

SFR Page = Normal

SFR Address = 0xD6 RESET = 0000-0000

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| P37KBIE | P36KBIE | P35KBIE | P34KBIE | **P41KBIE** | **P40KBIE** | P31KBIE | P30KBIE |
| **R/W** | **R/W** | **R/W** | **R/W** | **R/W** | **R/W** | **R/W** | **R/W** |

Bit 7~0: Keypad Input function enable bit for P3.7 ~ P3.4, P4.1, P4.0, P3.1 and P3.0 pins.
0: Disable associated port pin for keypad input function.
1: Enable associated port pin for keypad input function on low level.

## 11.4. Power Control Sample Code

*(1) Required function: Select Slow mode with OSCin/128 (default is OSCin/2)*

Assembly Code Example:

```
ORL     CKCON0,#(SCKS0 | SCKS1 | SCKS2)     ; OSCin/128

MOV     IFADRL,#DCON0                       ; Index Page-P address to DCON0
CALL    _page_p_sfr_read                    ; Read DCON0 data

ANL     IFD,#~(HSE)                         ; Disable HSE when SYSCLK ≤ 6MHz for power saving
CALL    _page_p_sfr_write                   ; Write data to DCON0
```

C Code Example:

```
CKCON0 |= (SCKS2 | SCKS1 | SCKS0);        // Select system clock divider to OSCin/128.

IFADRL = DCON0;                           // Index Page-P address to DCON0
page_p_sfr_read();                        // Read DCON0 data.

IFD &= ~HSE;                              // Disable HSE when SYSCLK ≤ 6MHz for power saving
page_p_sfr_write();                       // Write data to DCON0
```

*(2) Required function: Select Sub-Clock mode with OSCin/2 (default is OSCin/2=32KHz/2=16KHz)*

Assembly Code Example:

```
MOV     IFADRL,#CKCON2                    ; Index Page-P address to CKCON2
CALL    _page_p_sfr_read                  ; Read CKCON2 data

ANL     IFD,#~(OSCS1|OSCS0)               ; Switch OSCin source to ILRCO
ORL     IFD,#OSCS1
CALL    _page_p_sfr_write                 ; Write data to CKCON2

ANL     IFD,#~(IHRCOE|XTALE)              ; Disable IHRCO & XTAL
CALL    _page_p_sfr_write                 ; Write data to CKCON2

MOV     IFADRL,#DCON0)                    ; Index Page-P address to DCON0
CALL    _page_p_sfr_read                  ; Read DCON0 data

ANL     IFD,#~(HSE)                       ; Disable HSE when SYSCLK ≤ 6MHz for power saving
CALL    _page_p_sfr_write                 ; Write data to DCON0

MOV     A,CKCON0                          ; Select system clock = OSCin/2
ANL     A,#~(SCKS2|SCKS1|SCKS0)
ORL     A,#SCKS0
MOV     CKCON0,A
```

C Code Example:

```
IFADRL = CKCON2;                          // Index Page-P address to CKCON2
page_p_sfr_read();                        // Read CKCON2 data

IFD &= ~(OSCS1 | OSCS0);                  // Switch OSCin source to ILRCO
IFD |= OSCS1;
page_p_sfr_write();                       // Write data to CKCON2

IFD = IFD & ~(IHRCOE|XTALE);              // Disable IHRCO & XTAL
page_p_sfr_write();                       // Write data to CKCON2

IFADRL = DCON0;                           // Index Page-P address to DCON0
page_p_sfr_read();                        // Read DCON0 data

IFD = IFD & ~(HSE);                       // Disable HSE when SYSCLK ≤ 6MHz for power saving
```

```
    page_p_sfr_write();                     // Write data to DCON0

    ACC = CKCON0;                           // Select system clock = OSCin/2
    ACC &= ~(SCKS2 | SCKS1 | SCKS0);
    ACC |= SCKS0;
    CKCON0 = ACC;
```

*(3). Required Function: Switch MCU running with XTAL mode*

Assembly Code Example:

```
    MOV     IFADRL,#CKCON2              ; Index Page-P address to CKCON2
    CALL    _page_p_sfr_read            ; Read CKCON2 data

    ORL     IFD,#(XTALE)                ; Enable XTAL oscillating
    CALL    _page_p_sfr_write           ; Write data to CKCON2

check_XTOR_0:                           ; Check XTAL oscillating ready
    MOV     A,AUXR1
    JNB     ACC.4,check_XTOR_0          ; Waiting for XTOR(AUXR1.4) true

    ANL     IFD,#~(OSCS1|OSCS0)         ; Switch OSCin source to XTAL
    ORL     IFD,#OSCS0
    CALL    _page_p_sfr_write           ; Write data to CKCON2

    ANL     IFD,#~(IHRCOE)              ; Disable IHRCO
    CALL    _page_p_sfr_write           ; Write data to CKCON2

    MOV     IFADRL,#DCON0               ; Index Page-P address to DCON0
    CALL    _page_p_sfr_read            ; Read DCON0 data

    ANL     IFD,#~(HSE)                 ; Disable HSE when SYSCLK ≤ 6MHz for power saving
    CALL    _page_p_sfr_write           ; Write data to DCON0

    ANL     CKCON0,#~(SCKS2|SCKS1|SCKS0) ; SYSCLK = OSCin/1
```

C Code Example:

```
    IFADRL = CKCON2;                    // Index Page-P address to CKCON2
    page_p_sfr_read();                  // Read CKCON2 data

    IFD |= XTALE;                       // Enable XTAL oscillating
    page_p_sfr_write();                 // Write data to CKCON2

    while( AUXR1&XTOR == 0x00 );        // Check XTAL oscillating ready
                                        // Waiting for XTOR(AUXR1.4) true

    IFD &= ~(OSCS1 | OSCS0);            // Switch OSCin source to XTAL.
    IFD |= OSCS0;
    page_p_sfr_write ();                // Write data to CKCON2

    IFD &= ~IHRCOE;                     // Disable IHRCO if MCU is switched from IHRCO
    page_p_sfr_write();                 // Write data to CKCON2.

    IFADRL = DCON0;                     // Index Page-P address to DCON0
    page_p_sfr_read();                  // Read DCON0 data.

    IFD &= ~HSE;                        // Disable HSE when SYSCLK ≤ 6MHz for power saving
    page_p_sfr_write();                 // Write data to DCON0

    CKCON0 &= ~(SCKS2 | SCKS1 | SCKS0);  // SYSCLK = OSCin/1
```

*(4). Required Function: Enter Watch mode with 2S wake-up duration*

Assembly Code Example:

```
    ORG     0003Bh
SystemFlag_ISR:
```

```
    ANL     PCON1,#(WDTF)                   ; Clear WDT flag (write “1”)
    RETI

main:
    ANL     PCON1,#WDTF                     ; Clear WDTF flag (write “1”)
    ORL     WDTCR,#(NSW|ENW|PS2|PS1|PS0)
                                            ;Enable WDT and NSW (for watch mode)
                                            ;Set PS[2:0] = 7 to select WDT period for 1.984s

    ORL     SFIE,#WDTFIE                    ; Enable WDT interrupt
    ORL     EIE1,#ESF                       ; Enable SystemFlag interrupt
    SETB    EA                              ; Enable Global interrupt

    ORL     PCON0,#PD                       ; Set MCU to power down

;   MCU wait for wake-up
```

C Code Example:

```
void SystemFlag_ISR (void) interrupt 7
{
    PCON1 &= WDTF;                          // Clear WDT flag (write “1”)
}

viod main (void)
{
    PCON1 &= WDTF;                          // Clear WDT flag (write “1”)
    WDTCR |= (NSW | ENW | PS2 | PS1 | PS0); // Enable WDT and NSW (for watch mode)
                                            // Set PS[2:0] = 7 to select WDT period for 1.984s

    SFIE |= WDTFIE;                         // Enable WDT interrupt
    EIE1 |= ESF;                            // Enable SystemFlag interrupt
    EA = 1;                                 // Enable global interrupt

    PCON0 |= PD;                            // Set MCU to power down

//  MCU wait for wake-up
}
```

*(5). Required Function: Monitor Mode (L-series only)*

Assembly Code Example:

```
    ORG     0003Bh
SystemFlag_ISR:
    ANL     PCON1,#(BOF0)                   ; Clear BOD0 flag (write “1”)
    RETI

main:
    MOV     IFADRL,#PCON2                   ; Index Page-P address to PCON2
    CALL    _page_p_sfr_read                ; Read PCON2 data

    ORL     IFD,#AWBOD0                     ; Enable BOD0 operating in power-down mode
    CALL    _page_p_sfr_write               ; Write data to PCON2

    ORL     SFIE,#BOF0IE                    ; Enable BOF0 interrupt
    ORL     EIE1,#ESF                       ; Enable SystemFlag interrupt
    SETB    EA                              ; Enable global interrupt

    ORL     PCON0,#PD                       ; Set MCU to power down

;   MCU wait for wake-up
```

C Code Example:

```
void SystemFlag_ISR() interrupt 7
{
```

```
    PCON1 &= BOF0;                          // Clear BOD0 flag (write “1”)
}

void main()
{
    IFADRL = PCON2;                         // Index Page-P address to PCON2
    page_p_sfr_read();                      // Read PCON2 data

    IFD |= AWBOD0;                          // Enable BOD0 operating in power-down mode
    page_p_sfr_write();                     // Write data to PCON2


    SFIE |= BOF0IE;                         // Enable BOD0 interrupt
    EIE1 |= ESF;                            // Enable SystemFlag interrupt
    EA = 1;                                 // Enable global interrupt

    PCON0 |= PD;                            // Set MCU to power down

//  MCU wait for wake-up
}
```

*(6). Required Function: Safety and Fast wakeup for XTAL mode in power-down*

Assembly Code Example:

```
    MOV     IFADRL,#CKCON2              ; Index Page-P address to CKCON2
    CALL    _page_p_sfr_read            ; Read CKCON2 data

    ORL     IFD,#IHRCOE                 ; Enable IHRCO
    CALL    _page_p_sfr_write           ; Write data to CKCON2

    Delay_32us

    ANL     IFD,#~(OSCS1|OSCS0)         ; Switch OSCin source to IHRCO
    CALL    _page_p_sfr_write           ; Write data to CKCON2

    ORL     PCON0,#PD                   ; Set MCU to power down

;   MCU wait for wake-up

check_XTOR:                             ; Check XTAL oscillating ready
    MOV     A,AUXR1
    JNB     ACC.4,check_XTOR            ; Waiting for XTOR(AUXR1.4) true

    ANL     IFD,#~(OSCS1 | OSCS0)       ; Switch OSCin source to XTAL.
    ORL     IFD,#(OSCS0)
    CALL    _page_p_sfr_write           ; Write data to CKCON2

    ANL     IFD,#~(IHRCOE)              ; Disable IHRCO if MCU is switched from IHRCO
    CALL    _page_p_sfr_write           ; Write data to CKCON2
```

C Code Example:

```
    IFADRL = CKCON2;                    // Index Page-P address to CKCON2
    page_p_sfr_read();                  // Read CKCON2 data

    IFD |= IHRCOE;                      // Enable IHRCO
    page_p_sfr_write();                 // Write data to CKCON2

    Delay_32us();

    IFD &= ~(OSCS1 | OSCS0);            // Switch OSCin source to IHRCO
    page_p_sfr_write();                 // Write data to CKCON2

    PCON0 |= PD;                        // Set MCU to power down

//  MCU wait for wake-up
```

```
while(AUXR1 & XTOR == 0x00);          // Check XTAL oscillating ready
                                      // Waiting for XTOR(AUXR1.4) true

IFD &= ~(OSCS1 | OSCS0);              // Switch OSCin source to XTAL.
IFD |= OSCS0;
page_p_sfr_write ();                  // Write data to CKCON2

IFD &= ~IHRCOE;                       // Disable IHRCO if MCU is switched from IHRCO
page_p_sfr_write();                   // Write data to CKCON2.
```

# 12. Configurable I/O Ports

The **MG86FE/L104** has following I/O ports: P1.0~P1.7, P3.0~P3.5 and P3.7. RST pin has a swapped function for P3.6. If select external crystal oscillator as system clock input, Port 4.0 and Port 4.1 are configured to XTAL2 and XTAL1. The exact number of I/O pins available depends upon the package types. See Table 12–1.

Table 12–1. Number of I/O Pins Available

| Package Type | I/O Pins | Number of I/O ports |
|---|---|---|
| 20-pin SOP | P1.0~P1.7, P3.0~P3.5, P3.7,<br>RST(P3.6), XTAL2(P4.0), XTAL1(P4.1) | 15 or 16 (RST/P3.6) or<br>18 (INTOSC enabled) |
| 16-pin SOP | P1.0~P1.6, P3.0~P3.3,<br>RST(P3.6), XTAL2(P4.0), XTAL1(P4.1) | 11 or 12 (RST/P3.6) or<br>14 (INTOSC enabled) |
| 8-pin SOP | P3.0~P3.5 | 6 |

## 12.1. IO Structure

The I/O operating modes are distinguished two groups in **MG86FE/L104**. The first group is only for Port 3 to support four configurations on I/O operating. These are: quasi-bidirectional (standard 8051 I/O port), push-pull output, input-only (high-impedance input) and open-drain output.

All other general port pins are belonged to secondary group. They can be configured to two output modes, push-pull output and open-drain output with pull-up resistor control.

Followings describe the configuration of the all types I/O mode.

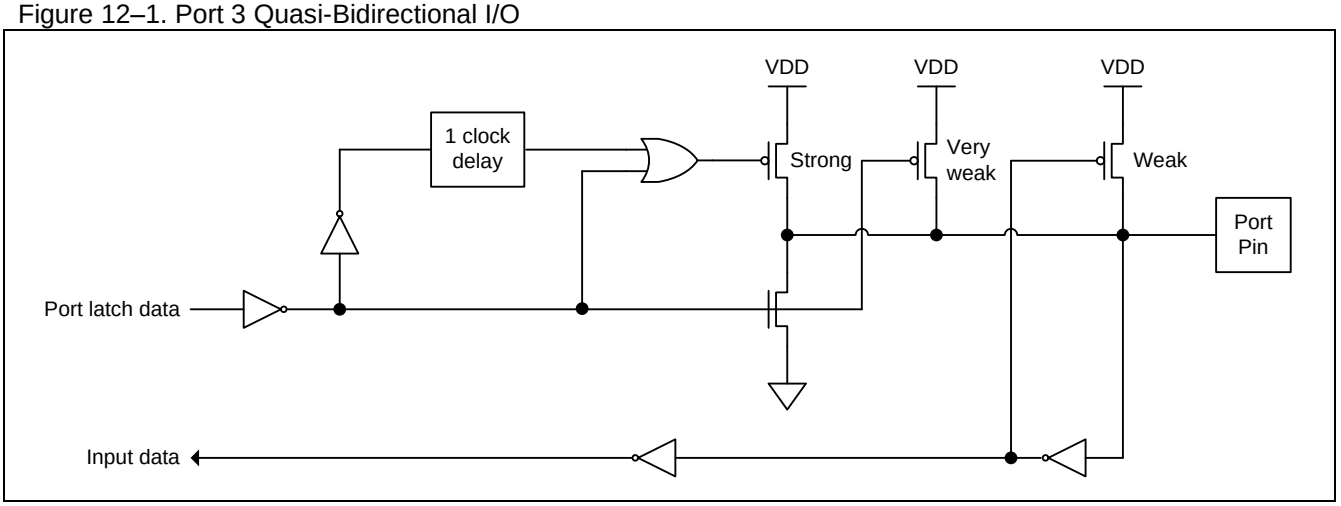
### 12.1.1. Port 3 Quasi-Bidirectional IO Structure

Port 3 pins in quasi-bidirectional mode are similar to the standard 8051 port pins. A quasi-bidirectional port can be used as an input and output without the need to reconfigure the port. This is possible because when the port outputs a logic high, it is weakly driven, allowing an external device to pull the pin low. When the pin outputs low, it is driven strongly and able to sink a large current. There are three pull-up transistors in the quasi-bidirectional output that serve different purposes.

One of these pull-ups, called the “very weak” pull-up, is turned on whenever the port register for the pin contains a logic “1”. This very weak pull-up sources a very small current that will pull the pin high if it is left floating. A second pull-up, called the “weak” pull-up, is turned on when the port register for the pin contains a logic “1” and the pin itself is also at a logic “1” level. This pull-up provides the primary source current for a quasi-bidirectional pin that is outputting a 1. If this pin is pulled low by the external device, this weak pull-up turns off, and only the very weak pull-up remains on. In order to pull the pin low under these conditions, the external device has to sink enough current to over-power the weak pull-up and pull the port pin below its input threshold voltage. The third pull-up is referred to as the “strong” pull-up. This pull-up is used to speed up low-to-high transitions on a quasi-bidirectional port pin when the port register changes from a logic “0” to a logic “1”. When this occurs, the strong pull-up turns on for one CPU clocks, quickly pulling the port pin high.

The quasi-bidirectional port configuration on Port 3 is shown in Figure 12–1.

Figure 12–1. Port 3 Quasi-Bidirectional I/O

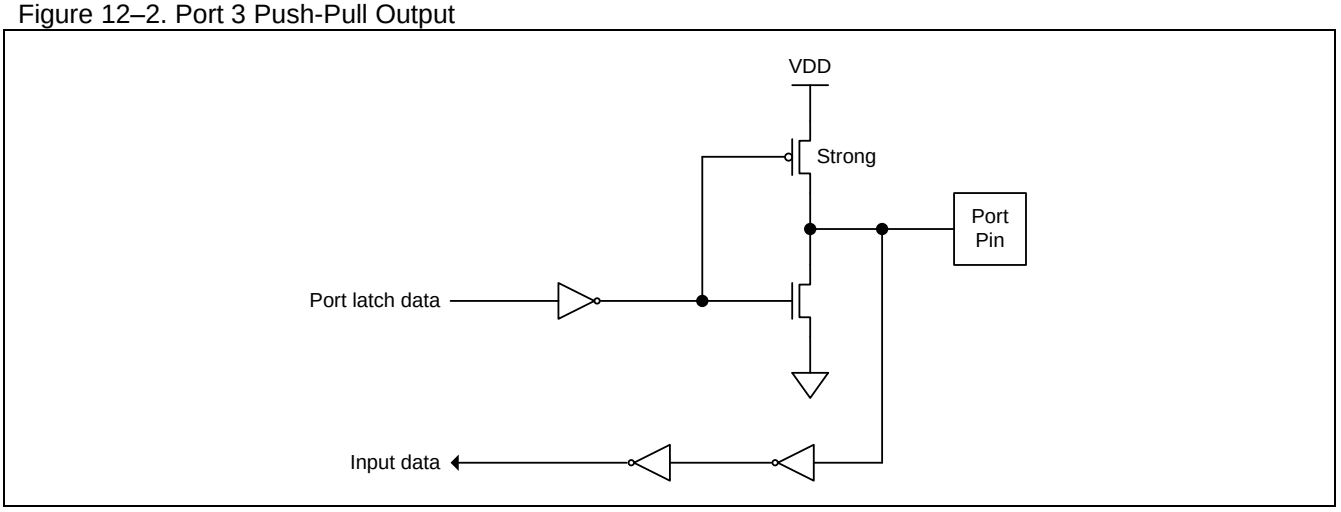



### 12.1.2. Port 3 Push-Pull Output Structure

The push-pull output configuration on Port 3 has the same pull-down structure as both the open-drain and the quasi-bidirectional output modes, but provides a continuous strong pull-up when the port register contains a logic "1". The push-pull mode may be used when more source current is needed from a port output. In addition, the input path of the port pin in this configuration is also the same as quasi-bidirectional mode.

The push-pull port configuration on Port 3 is shown in Figure 12–2.

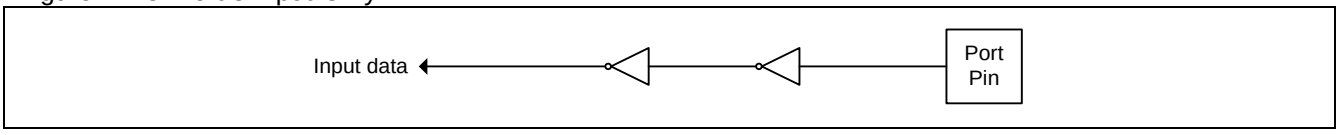
Figure 12–2. Port 3 Push-Pull Output




### 12.1.3. Port 3 Input-Only (High Impedance Input) Structure

The input-only configuration on Port 3 is an input without any pull-up resistors on the pin, as shown in Figure 12–3.

Figure 12–3. Port 3 Input-Only

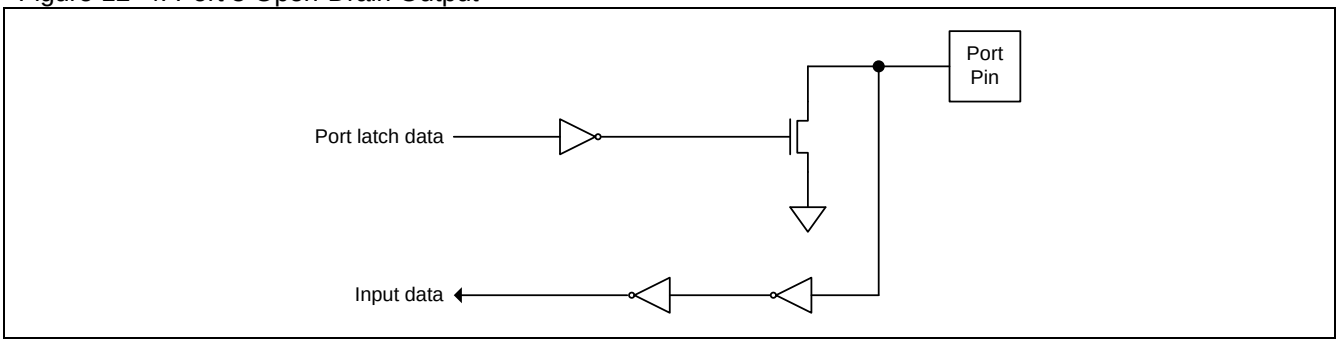


### 12.1.4. Port 3 Open-Drain Output Structure

The open-drain output configuration on Port 3 turns off all pull-ups and only drives the pull-down transistor of the port pin when the port register contains a logic “0”. To use this configuration in application, a port pin must have an external pull-up, typically a resistor tied to VDD. The pull-down for this mode is the same as for the quasi-bidirectional mode. In addition, the input path of the port pin in this configuration is also the same as quasi-bidirectional mode.

The Port 3 open-drain port configuration is shown in Figure 12–4.

Figure 12–4. Port 3 Open-Drain Output

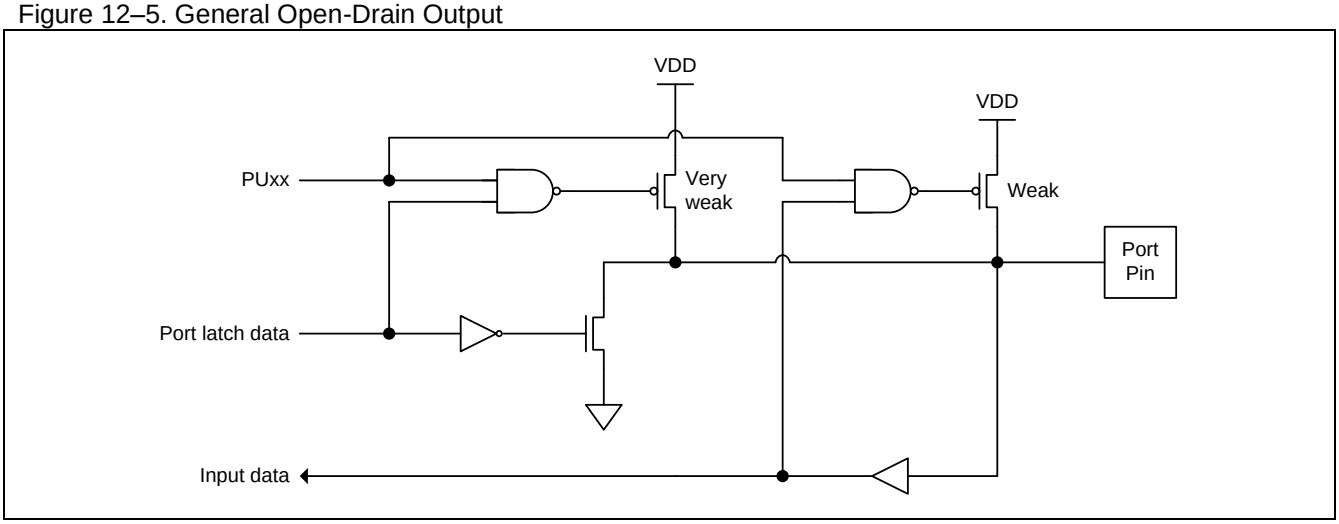



### 12.1.5. General Open-Drain Output Structure

The open-drain output configuration on general port pins only drives the pull-down transistor of the port pin when the Port Data register contains a logic “0”. To use this configuration in application, a port pin can select an external pull-up, or an on-chip pull-up by software enabled in PUCON0.

The general open-drain port configuration is shown in Figure 12–5.

Figure 12–5. General Open-Drain Output

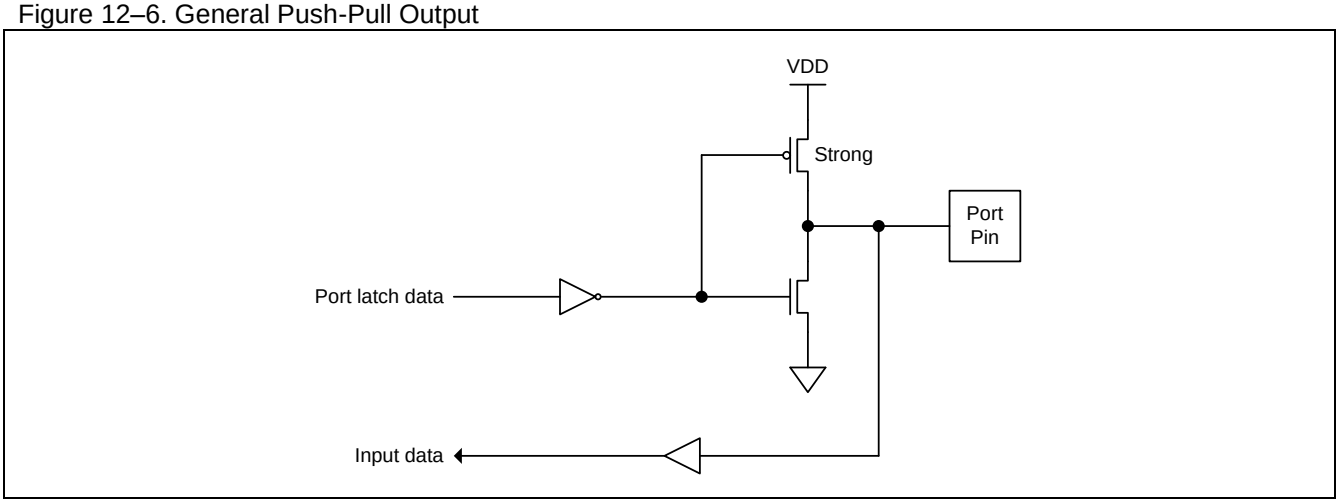



### 12.1.6. General Push-Pull Output Structure

The push-pull output configuration on general port pins has the same pull-down structure as the open-drain output modes, but provides a continuous strong pull-up when the port register contains a logic “1”. The push-pull mode may be used when more source current is needed from a port output. In addition, the input path of the port pin in this configuration is also the same as open-drain mode.

The push-pull port configuration is shown in Figure 12–6.

Figure 12–6. General Push-Pull Output




### 12.1.7. General Port Input Configured

A Port pin is configured as a digital input by setting its output mode to “Open-Drain” and writing a logic “1” to the associated bit in the Port Data register. For example, P1.7 is configured as a digital input by setting P1M0.7 to a logic 0 and P1.7 to a logic 1.

## 12.2. I/O Port Register

All I/O port pins on the **MG86FE/L104** may be individually and independently configured by software to select its operating mode. Only Port 3 has four operating modes, as shown in Table 12–2. Two mode registers select the output type for each port 3 pin.

Table 12–2. Port 3 Configuration Settings

| P3M0.y | P3M1.y | Port Mode |
|---|---|---|
| 0 | 0 | Quasi-Bidirectional |
| 0 | 1 | Push-Pull Output |
| 1 | 0 | Input Only (High Impedance Input) |
| 1 | 1 | Open-Drain Output |

Where y=0~7 (port pin). The registers P3M0 and P3M1 are listed in each port description.

Other general port pins support two operating modes, as shown in Table 12–3. One mode register selects the output type for each port pin.

Table 12–3. General Port Configuration Settings

| PxM0.y | Port Mode |
|---|---|
| 0 | Open-Drain Output |
| 1 | Push-Pull Output |

Where x=1, 4 (port number), and y=0~7 (port pin). The registers PxM0 and PxM1 are listed in each port description.

### 12.2.1. Port 1 Register

***P1: Port 1 Register***

SFR Page = Normal
SFR Address = 0x90 RESET = 1111-1111

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| P1.7 | P1.6 | P1.5 | P1.4 | P1.3 | P1.2 | P1.1 | P1.0 |
| **R/W** | **R/W** | **R/W** | **R/W** | **R/W** | **R/W** | **R/W** | **R/W** |

Bit 7~0: P1.7~P1.0 could be only set/cleared by CPU.

***P1M0: Port 1 Mode Register 0***

SFR Page = Normal
SFR Address = 0x91 RESET = 0000-0000

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| P1M0.7 | P1M0.6 | P1M0.5 | P1M0.4 | P1M0.3 | P1M0.2 | P1M0.1 | P1M0.0 |
| **R/W** | **R/W** | **R/W** | **R/W** | **R/W** | **R/W** | **R/W** | **R/W** |

0: Port pin output mode is configured to open-drain.
1: Port pin output mode is configured to push-pull.

### 12.2.2. Port 3 Register

***P3: Port 3 Register***

SFR Page = Normal
SFR Address = 0xB0 RESET = 1111-1111

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| P3.7 | P3.6 | P3.5 | P3.4 | P3.3 | P3.2 | P3.1 | P3.0 |
| **R/W** | **R/W** | **R/W** | **R/W** | **R/W** | **R/W** | **R/W** | **R/W** |

Bit 7~0: P3.7~P3.0 could be only set/cleared by CPU.

***P3M0: Port 3 Mode Register 0***

SFR Page = Normal
SFR Address = 0xB1 RESET = 0000-0000

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| P3M0.7 | P3M0.6 | P3M0.5 | P3M0.4 | P3M0.3 | P3M0.2 | P3M0.1 | P3M0.0 |
| **R/W** | **R/W** | **R/W** | **R/W** | **R/W** | **R/W** | **R/W** | **R/W** |

***P3M1: Port 3 Mode Register 1***

SFR Page = Normal
SFR Address = 0xB2 RESET = 0000-0000

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| P3M1.7 | P3M1.6 | P3M1.5 | P3M1.4 | P3M1.3 | P3M1.2 | P3M1.1 | P3M1.0 |
| **R/W** | **R/W** | **R/W** | **R/W** | **R/W** | **R/W** | **R/W** | **R/W** |

### 12.2.3. Port 4 Register

***P4: Port 4 Register***

SFR Page = Normal
SFR Address = 0xE8 RESET = xxxx-xx11

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| -- | -- | -- | -- | -- | -- | P4.1 | P4.0 |
| **W** | **W** | **W** | **W** | **W** | **W** | **R/W** | **R/W** |

Bit 7~2: Reserved.

Bit 1~0: P4.1~P4.0 could be only set/cleared by CPU. P4.1 and P4.0 have the alternated function for crystal oscillating circuit, XTAL1 and XTAL2.

***P4M0: Port 4 Mode Register 0***

SFR Page = Normal
SFR Address = 0xB3 RESET = xxxx-xx00

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| -- | -- | -- | -- | -- | -- | P4M0.1 | P4M0.0 |
| **W** | **W** | **W** | **W** | **W** | **W** | **R/W** | **R/W** |

0: Port pin output mode is configured to open-drain.
1: Port pin output mode is configured to push-pull.

### 12.2.4. Pull-Up Control Register

***PUCON0: Port Pull-up Control Register 0***

SFR Page = Normal
SFR Address = 0xB4 RESET = x0xx-00xx

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| -- | PU40 | -- | -- | PU11 | PU10 | -- | -- |
| **W** | **R/W** | **W** | **W** | **R/W** | **R/W** | **W** | **W** |

Bit 7: Reserved. Software must write “0” on this bit when PUCON0 is written.

Bit 6: Port 4 pull-up enable control on low nibble.
0: Disable the P4.0 & P4.1 pull-up resistor in open-drain output mode.
1: Enable the P4.0 & P4.1 pull-up resistor in open-drain output mode.

Bit 5~4: Reserved. Software must write “0” on these bits when PUCON0 is written.

Bit 3: Port 1 pull-up enable control on high nibble.
0: Disable the P1.7 ~ P1.4 pull-up resistor in open-drain output mode.
1: Enable the P1.7 ~ P1.4 pull-up resistor in open-drain output mode.

Bit 2: Port 1 pull-up enable control on low nibble.
0: Disable the P1.3 ~ P1.0 pull-up resistor in open-drain output mode.
1: Enable the P1.3 ~ P1.0 pull-up resistor in open-drain output mode.

Bit 1~0: Reserved. Software must write “0” on these bits when PUCON0 is written.

## 12.3. GPIO Sample Code

*(1). Required Function: Set P1.0 to input mode with on-chip pull-up resistor enabled*

Assembly Code Example:

```
ANL     P1M0,#~P1M00            ; Configure P1.0 to open drain mode
SETB    P10                     ; Set P1.0 data latch to “1” to enable input mode
ORL     PUCON0,#PU10            ; Enable the P1.3~P1.0 on-chip pull-up resistor
```

C Code Example:

```
P1M0 &= P1M00;                  // Configure P1.0 to open drain mode
P10 = 1;                        // Set P1.0 data latch to “1” to enable input mode
PUCON0 |= PU10;                 // Enable the P1.3~P1.0 on-chip pull-up resistor
```

*(2). Required Function: Switch RST pint to P3.6*

**Assembly Code Example:**

```
MOV     IFADRL,#DCON0           ; Index Page-P address to DCON0
CALL    _page_p_sfr_read        ; Read DCON0 data

ANL     IFD,#~(RSTIO)           ; Select I/O pad function for P36
CALL    _page_p_sfr_write       ; Write data to DCON0
```

**C Code Example:**

```
IFADRL = DCON0;                 // Index Page-P address to DCON0
page_p_sfr_read();              // Read DCON0 data.

IFD &= ~RSTIO;                  // Select I/O pad function for P36
page_p_sfr_write();             // Write data to DCON0
```

# 13. Interrupt

The **MG86FE/L104** has 6 interrupt sources with a four-level interrupt structure. There are several SFRs associated with the four-level interrupt. They are the IE, IP0L, IP0H, EIE1, EIP1L and EIP1H. The IP0H (Interrupt Priority 0 High) and EIP1H (Extended Interrupt Priority 1 High) registers make the four-level interrupt structure possible. The four priority level interrupt structure allows great flexibility in handling these interrupt sources.

## 13.1. Interrupt Structure

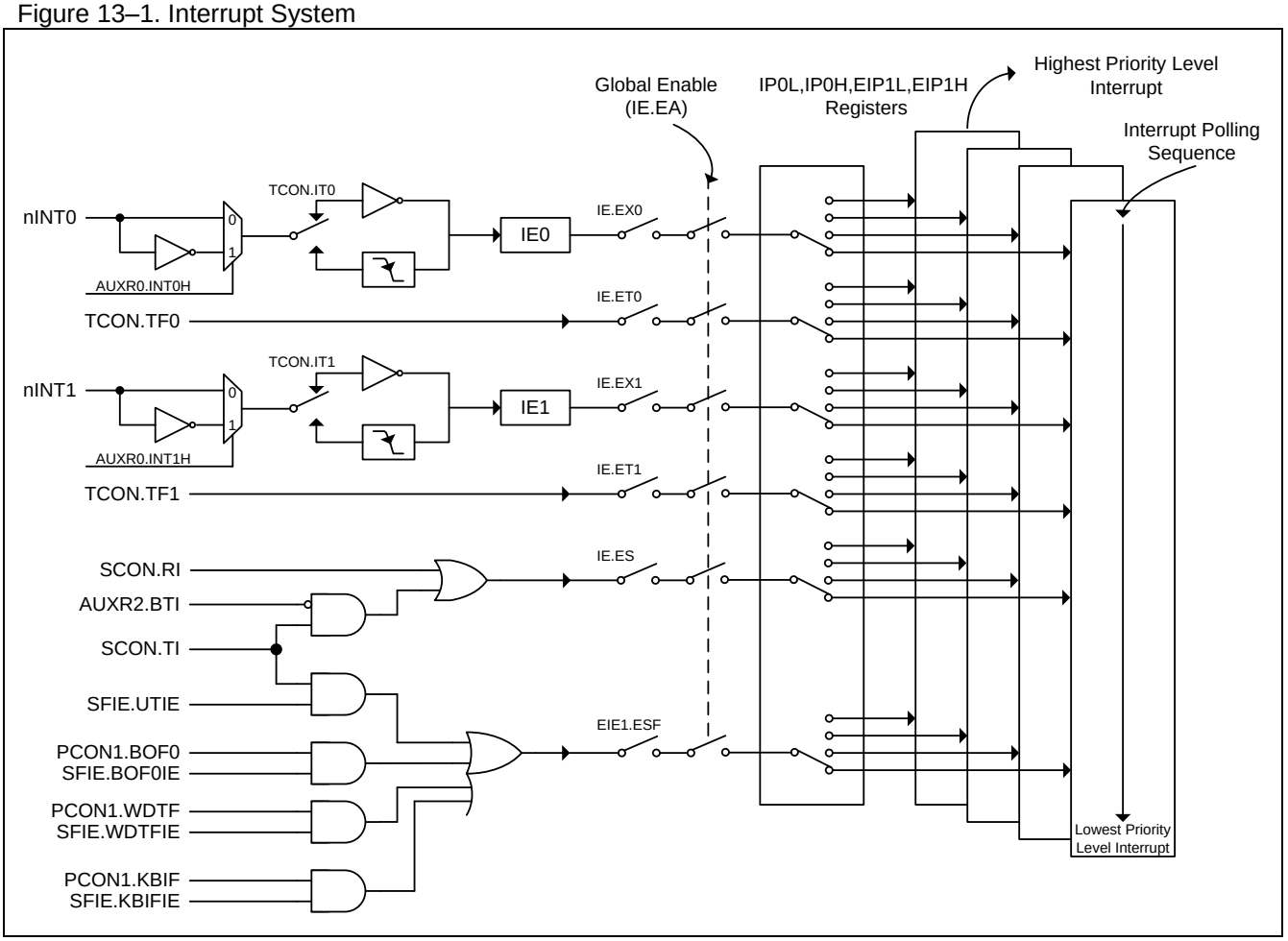
Table 13–1 lists all the interrupt sources. The 'Request Bits' are the interrupt flags that will generate an interrupt if it is enabled by setting the 'Enable Bit'. Of course, the global enable bit EA (in IE0 register) should have been set previously. The 'Request Bits' can be set or cleared by software, with the same result as though it had been set or cleared by hardware. That is, interrupts can be generated or pending interrupts can be cancelled in software. The 'Priority Bits' determine the priority level for each interrupt. The 'Priority within Level' is the polling sequence used to resolve simultaneous requests of the same priority level. The 'Vector Address' is the entry point of an interrupt service routine in the program memory.

Figure 13–1 shows the interrupt system. Each of these interrupts will be briefly described in the following sections.

Table 13–1. Interrupt Sources

| No | Source Name | Enable Bit | Request Bits | Priority Bits | Polling Priority | Vector Address |
|---|---|---|---|---|---|---|
| #1 | External Interrupt 0, nINT0 | EX0 | IE0 | [ PX0H, PX0L ] | (Highest) | 0003H |
| #2 | Timer 0 | ET0 | TF0 | [ PT0H, PT0L ] | … | 000Bh |
| #3 | External Interrupt 1, nINT1 | EX1 | IE1 | [ PX1H, PX1L ] | ... | 0013H |
| #4 | Timer 1 | ET1 | TF1 | [ PT1H, PT1L ] | … | 001BH |
| #5 | Serial Port 0 | ES | RI, TI | [ PSH, PSL ] | … | 0023H |
| #6 | System Flag | ESF | BOF0,WDTF KBIF,(TI) | [ PSFH, PSFL ] | (Lowest) | 002BH |

Figure 13–1. Interrupt System




## 13.2. Interrupt Source

Table 13–2. Interrupt Source Flag

| No | Source Name | Request Bits | Bit Location |
|---|---|---|---|
| #1 | External Interrupt, nINT0 | IE0 | TCON.1 |
| #2 | Timer 0 | TF0 | TCON.5 |
| #3 | External Interrupt, nINT1 | IE1 | TCON.3 |
| #4 | Timer 1 | TF1 | TCON.7 |
| #5 | Serial Port 0 | RI0<br>TI0 | SCON0.0<br>SCON0.1 |
| #6 | System Flag | WDTF<br>BOF0<br>KBIF<br>(TI) | PCON1.0<br>PCON1.1<br>PCON1.3<br>PCON1.7 |

The external interrupt nINT0 and nINT1 can each be either level-activated or transition-activated, depending on bits IT0 and IT1 in register TCON. The flags that actually generate these interrupts are bits IE0 and IE1 in TCON. When an external interrupt is generated, the flag that generated it is cleared by the hardware when the service routine is vectored to *only if the interrupt was transition –activated*, then the external requesting source is what controls the request flag, rather than the on-chip hardware.

The Timer0 and Timer1 interrupts are generated by TF0 and TF1, which are set by a rollover in their respective Timer/Counter registers in most cases. When a timer interrupt is generated, the flag that generated it is cleared by the on-chip hardware when the service routine is vectored to.

The serial port interrupt is generated by the logical OR of RI and TI. Neither of these flags is cleared by hardware when the service routine is vectored to. The service routine should poll RI and TI to determine which one to request service and it will be cleared by software.

The System Flag interrupt is generated by KBIF, BOF0 and WDTF in PCON1. KBIF is set by KBI event. BOF0 is set by on chip Brownout-Detector (BOD0) met the low voltage event. WDTF is set by Watch-Dog-Timer overflow. They will not be cleared by hardware when the service routine is vectored to.
*Note: the WDTFIE function is under verifying.*

All of the bits that generate interrupts can be set or cleared by software, with the same result as though it had been set or cleared by hardware. In other words, interrupts can be generated or pending interrupts can be canceled in software.

## 13.3. Interrupt Enable

Table 13–3. Interrupt Enable

| No | Source Name | Enable Bit | Bit Location |
|---|---|---|---|
| #1 | External Interrupt, nINT0 | EX0 | IE.0 |
| #2 | Timer 0 | ET0 | IE.1 |
| #3 | External Interrupt, nINT1 | EX1 | IE.2 |
| #4 | Timer 1 | ET1 | IE.3 |
| #5 | Serial Port 0 | ES0 | IE.4 |
| #6 | System Flag | ESF & (UTIE, KBIFIE, BOF0IE, WDTFIE) | EIE1.3 & SFIE.7,3,1,0 |

There are **6** interrupt sources available in **MG86FE/L104**. Each of these interrupt sources can be individually enabled or disabled by setting or clearing an interrupt enable bit in the registers IE and EIE1. IE also contains a global disable bit, EA, which can be cleared to disable all interrupts at once. If EA is set to '1', the interrupts are individually enabled or disabled by their corresponding enable bits. If EA is cleared to '0', all interrupts are disabled.

## 13.4. Interrupt Priority

The priority scheme for servicing the interrupts is the same as that for the 80C51, except there are four interrupt levels rather than two as on the 80C51. The Priority Bits (see Table 13–1) determine the priority level of each interrupt. IP0L, IP0H, EIP1L and EIP1H are combined to 4-level priority interrupt. Table 13–4 shows the bit values and priority levels associated with each combination.

Table 13–4. Interrupt Priority

| {IPnH.x , IPnL.x} | Priority Level |
|---|---|
| 11 | 1 (highest) |
| 10 | 2 |
| 01 | 3 |
| 00 | 4 |

Each interrupt source has two corresponding bits to represent its priority. One is located in SFR named IPnH and the other in IPnL register. Higher-priority interrupt will be not interrupted by lower-priority interrupt request. If two interrupt requests of different priority levels are received simultaneously, the request of higher priority is serviced. If interrupt requests of the same priority level are received simultaneously, an internal polling sequence determine which request is serviced. Table 13–2 shows the internal polling sequence in the same priority level and the interrupt vector address.

## 13.5. Interrupt Process

Each interrupt flag is sampled at every system clock cycle. The samples are polled during the next system clock. If one of the flags was in a set condition at first cycle, the second cycle(polling cycle) will find it and the interrupt system will generate an hardware LCALL to the appropriate service routine as long as it is not blocked by any of the following conditions.

Block conditions:

- An interrupt of equal or higher priority level is already in progress.
- The current cycle (polling cycle) is not the final cycle in the execution of the instruction in progress.
- The instruction in progress is RETI or any write to the IE, IP0L, IPH, EIE1, EIP1L and EIP1H registers.

Any of these three conditions will block the generation of the hardware LCALL to the interrupt service routine. Condition 2 ensures that the instruction in progress will be completed before vectoring into any service routine. Condition 3 ensures that if the instruction in progress is RETI or any access to IE or IP, then at least one or more instruction will be executed before any interrupt is vectored to.

## 13.6. Special Interrupt Vector for TI

The serial port interrupt from TI flag can be masked by BTI (AUXR2.6). If BTI is set, set TI flag will not generate a serial port interrupt. The serial port interrupt only reflects the RI flag.

If UTIE (SFIE.7) is set, TI flag will be combined into System Flag Interrupt. In this mode, TI interrupt shares the interrupt vector with KBIF, BOF0 and WDTF in System Flag Interrupt.

## 13.7. Interrupt Register

***TCON: Timer/Counter Control Register***

SFR Page = Normal
SFR Address = 0x88 RESET = 0000-0000

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| TF1 | TR1 | TF0 | TR0 | IE1 | IT1 | IE0 | IT0 |
| **R/W** | **R/W** | **R/W** | **R/W** | **R/W** | **R/W** | **R/W** | **R/W** |

Bit 3: IE1, Interrupt 1 Edge flag.
0: Cleared when interrupt processed on if transition-activated.
1: Set by hardware when external interrupt 1 edge is detected (transmitted or level-activated).

Bit 2: IT1: Interrupt 1 Type control bit.
0: Cleared by software to specify low level triggered external interrupt 1. If INT1H (AUXR0.1) is set, this bit specifies high level triggered on nINT1.
1: Set by software to specify falling edge triggered external interrupt 1. If INT1H (AUXR0.1) is set, this bit specifies rising edge triggered on nINT1.

Bit 1: IE0, Interrupt 0 Edge flag.
0: Cleared when interrupt processed on if transition-activated.
1: Set by hardware when external interrupt 0 edge is detected (transmitted or level-activated).

Bit 0: IT0: Interrupt 0 Type control bit.
0: Cleared by software to specify low level triggered external interrupt 0. If INT0H (AUXR0.0) is set, this bit specifies high level triggered on nINT0.
1: Set by software to specify falling edge triggered external interrupt 0. If INT0H (AUXR0.0) is set, this bit specifies rising edge triggered on nINT0.

***IE: Interrupt Enable Register***

SFR Page = Normal
SFR Address = 0xA8 RESET = 0xx0-0000

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| EA | -- | -- | ES | ET1 | EX1 | ET0 | EX0 |
| **R/W** | **W** | **W** | **R/W** | **R/W** | **R/W** | **R/W** | **R/W** |

Bit 7: EA, All interrupts enable register.
0: Global disables all interrupts.
1: Global enables all interrupts.

Bit 6: Reserved. Software must write “0” on this bit when IE is written.

Bit 5: Reserved. Software must write “0” on this bit when IE is written.

Bit 4: ES, Serial port interrupt enable register.
0: Disable serial port interrupt.
1: Enable serial port interrupt.

Bit 3: ET1, Timer 1 interrupt enable register.
0: Disable Timer 1 interrupt.
1: Enable Timer 1 interrupt.

Bit 2: EX1, External interrupt 1 enable register.
0: Disable external interrupt 1.
1: Enable external interrupt 1.

Bit 1: ET0, Timer 0 interrupt enable register.
0: Disable Timer 0 interrupt.
1: Enable Timer 1 interrupt.

Bit 0: EX0, External interrupt 0 enable register.
0: Disable external interrupt 0.
1: Enable external interrupt 1.

***EIE1: Extended Interrupt Enable 1 Register***

SFR Page = Normal
SFR Address = 0xAD RESET = xxxx-0xxx

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| -- | -- | -- | -- | ESF | -- | -- | -- |
| **W** | **W** | **W** | **W** | **R/W** | **W** | **W** | **W** |

Bit 7~4: Reserved. Software must write “0” on these bits when EIE1 is written.

Bit 3: ESF, Enable System Flag interrupt.
0: Disable the interrupt when the group of {KBIF, BOF0, WDTF} in PCON1 or TI in SCON is set
1: Enable the interrupt of the flags of {KBIF, BOF0, WDTF} in PCON1 or TI in SCON when the associated system flag interrupt is enabled in SFIE.

Bit 2~0: Reserved. Software must write “0” on these bits when EIE1 is written.

***SFIE: System Flag Interrupt Enable Register***

SFR Page = Normal
SFR Address = 0x8E RESET = 0xxx-0x00

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| UTIE | -- | -- | -- | KBIFIE | -- | BOF0IE | WDTFIE |
| **R/W** | **W** | **W** | **W** | **R/W** | **W** | **R/W** | **R/W** |

Bit 7: UART TI Enabled in system flag interrupt.
0: Disable the interrupt vector sharing for TI in system flag interrupt.
1: Set TI flag will share the interrupt vector with system flag interrupt.

Bit 6~4: Reserved. Software must write “0” on these bits when SFIE is written.

Bit 3: KBIFIE, Enable KBIF (PCON1.3) Interrupt.
0: Disable KBIF interrupt.
1: Enable KBIF interrupt.

Bit 2: Reserved. Software must write “0” on this bit when SFIE is written.

Bit 1: BOF0IE, Enable BOF0 (PCON1.1) Interrupt.
0: Disable BOF0 interrupt.
1: Enable BOF0 interrupt.

Bit 0: WDTFIE, Enable WDTF (PCON1.0) Interrupt.
0: Disable WDTF interrupt.
1: Enable WDTF interrupt.
*Note: the WDTFIE function is under verifying.*

***IP0L: Interrupt Priority 0 Low Register***

SFR Page = Normal
SFR Address = 0xB8 RESET = xxx0-0000

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| -- | -- | -- | PSL | PT1L | PX1L | PT0L | PX0L |
| **W** | **W** | **W** | **R/W** | **R/W** | **R/W** | **R/W** | **R/W** |

Bit 7~5: Reserved. Software must write “0” on these bits when IP0L is written.
Bit 4: PSL, Serial port interrupt priority-L register.
Bit 3: PT1L, Timer 1 interrupt priority-L register.
Bit 2: PX1L, external interrupt 1 priority-L register.
Bit 1: PT0L, Timer 0 interrupt priority-L register.
Bit 0: PX0L, external interrupt 0 priority-L register.

***IP0H: Interrupt Priority 0 High Register***

SFR Page = Normal

SFR Address = 0xB7 RESET = xxx0-0000

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| -- | -- | -- | PSH | PT1H | PX1H | PT0H | PX0H |
| **W** | **W** | **W** | **R/W** | **R/W** | **R/W** | **R/W** | **R/W** |

Bit 7~5: Reserved. Software must write “0” on these bits when IP0H is written.
Bit 4: PSH, Serial port interrupt priority-H register.
Bit 3: PT1H, Timer 1 interrupt priority-H register.
Bit 2: PX1H, external interrupt 1 priority-H register.
Bit 1: PT0H, Timer 0 interrupt priority-H register.
Bit 0: PX0H, external interrupt 0 priority-H register.

***EIP1L: Extended Interrupt Priority 1 Low Register***

SFR Page = Normal

SFR Address = 0xAE RESET = xxxx-0xxx

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| -- | -- | -- | -- | PSFL | -- | -- | -- |
| **W** | **W** | **W** | **W** | **R/W** | **W** | **W** | **W** |

Bit 7~4: Reserved. Software must write “0” on these bits when EIP1L is written.
Bit 3: PSFL, system flag interrupt priority-L register.
Bit 2~0: Reserved. Software must write “0” on these bits when EIP1L is written.

***EIP1H: Extended Interrupt Priority 1 High Register***

SFR Page = Normal

SFR Address = 0xAF RESET = xxxx-0xxx

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| -- | -- | -- | -- | PSFH | -- | -- | -- |
| **W** | **W** | **W** | **W** | **R/W** | **W** | **W** | **W** |

Bit 7~4: Reserved. Software must write “0” on these bits when EIP1H is written.
Bit 3: PSFH, system flag interrupt priority-H register.
Bit 2~0: Reserved. Software must write “0” on these bits when EIP1H is written.

***AUXR0: Auxiliary Register 0***

SFR Page = Normal

SFR Address = 0xA1 RESET = 000x-0000

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| P40OC1 | P40OC0 | P40FD | -- | P1FS1 | P1FS0 | **INT1H** | **INT0H** |
| **R/W** | **R/W** | **R/W** | **W** | **R/W** | **R/W** | **R/W** | **R/W** |

Bit 1: INT1H, INT1 High/Rising trigger enable.
0: Remain INT1 triggered on low level or falling edge on nINT1 port pin.
1: Set INT1 triggered on high level or rising edge on nINT1 port pin.

Bit 0: INT0H, INT0 High/Rising trigger enable.
0: Remain INT0 triggered on low level or falling edge on nINT0 port pin.
1: Set INT0 triggered on high level or rising edge on nINT0 port pin.

## 13.8. Interrupt Sample Code

*(1). Required Function: Set INT0 high level wake-up MCU in power-down mode*

Assembly Code Example:

```
      ORG       00003h
ext_int0_isr:
      to do.....
      RETI

main:

      SETB      P32                          ;

      ORL       IP0L,#PX0L                   ; Select INT0 interrupt priority
      ORL       IP0H,#PX0H                   ;

      ORL       AUXR0,#INT0H                 ; Set INT0 High level active

      JB        P32,$                        ; Confirm P3.2 input low

      SETB      EX0                          ; Enable INT0 interrupt
      CLR       IE0                          ; Clear INT0 flag
      SETB      EA                           ; Enable global interrupt

      ORL       PCON0,#PD                    ; Set MCU into Power Down mode
```

C Code Example:

```
void ext_int0_isr(void) interrupt 0
{
      To do……
}

void main(void)
{
      P32 = 1;

      IP0L |= PX0L;                          // Select INT0 interrupt priority
      IP0H |= PX0H;

      AUXR0 |= INT0H;                        // Set INT0 High level active

      while(P32);                            // Confirm P3.2 input low

      EX0 = 1;                               // Enable INT0 interrupt
      IE0 = 0;                               // Clear INT0 flag
      EA = 1;                                // Enable global interrupt

      PCON0 |= PD;                           // Set MCU into Power Down mode
}
```

# 14. Timers/Counters

**MG86FE/L104** has two Timers/Counters: Timer 0 and Timer 1. Timer0/1 can be configured as timers or event counters.

In the “timer” function, the timer rate is prescaled by 12 clock cycle to increment register value. In other words, it is to count the standard C51 machine cycle. AUXR2.T0X12 and AUXR2.T1X12 are the function for Timer 0/1 to set the timer rate on every clock cycle.

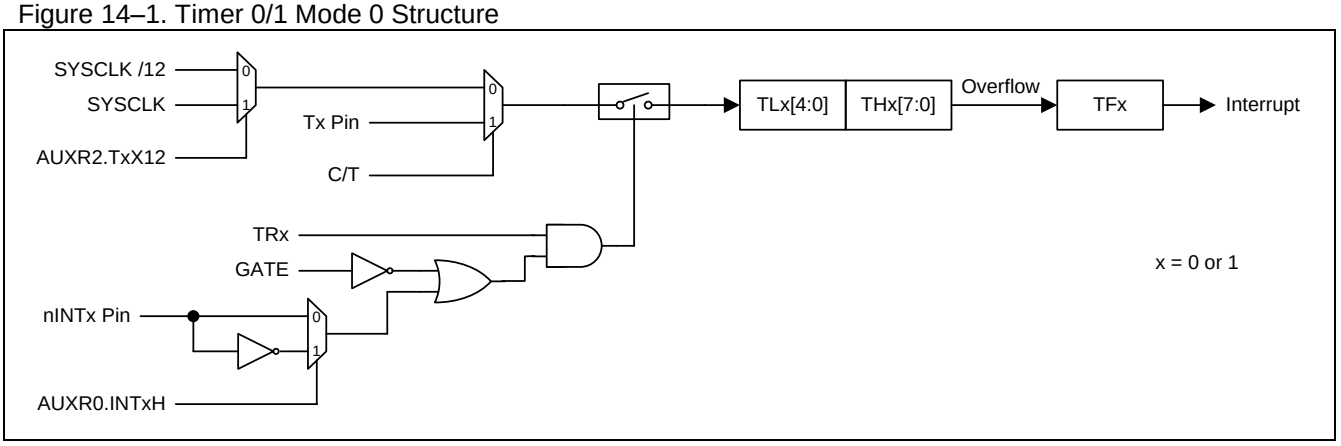
In the “counter” function, the register is incremented in response to a 1-to-0 transition at its corresponding external input pin, T0 or T1. In this function, the external input is sampled by every timer rate cycle. When the samples show a high in one cycle and a low in the next cycle, the count is incremented. The new count value appears in the register at the end of the cycle following the one in which the transition was detected.

## 14.1. Timer0 and Timer1

### 14.1.1. Mode 0 Structure

The timer register is configured as a 13-bit register. As the count rolls over from all 1s to all 0s, it sets the timer interrupt flag TFx. The counted input is enabled to the timer when TRx = 1 and either GATE=0 or INTx = 1. Mode 0 operation is the same for Timer0 and Timer1.

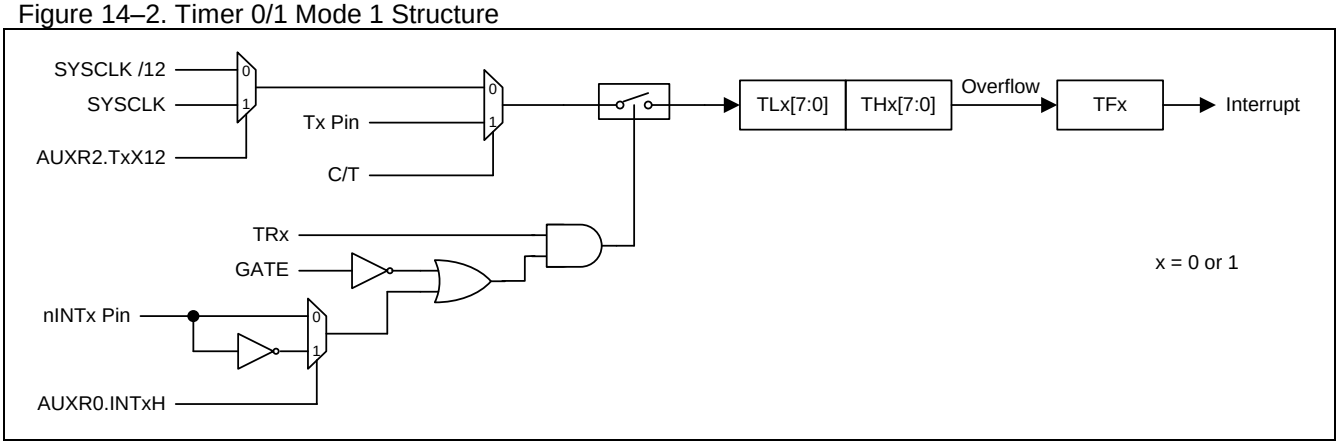
Figure 14–1. Timer 0/1 Mode 0 Structure



### 14.1.2. Mode 1 Structure

Mode1 is the same as Mode0, except that the timer register is being run with all 16 bits.

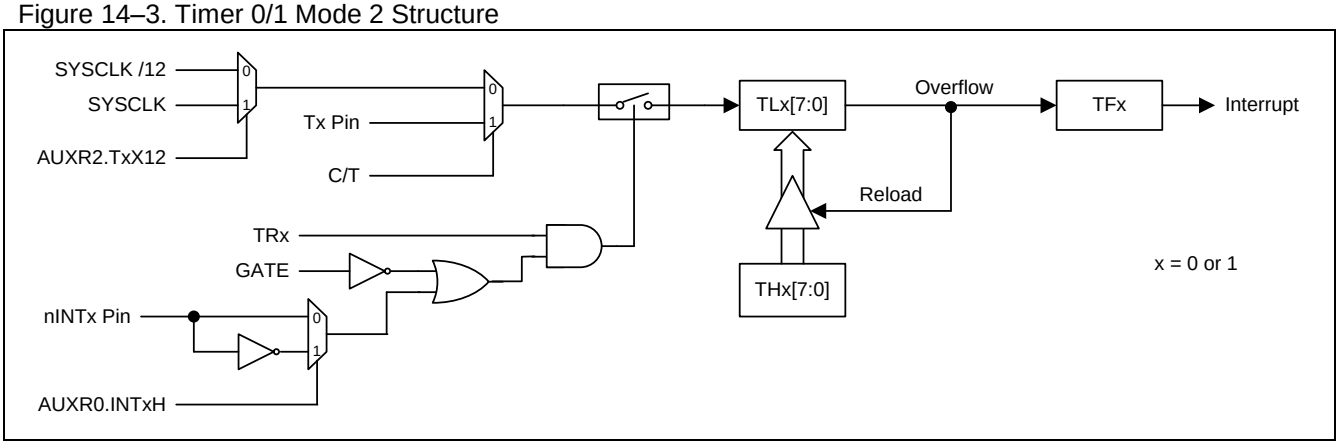
Figure 14–2. Timer 0/1 Mode 1 Structure




### 14.1.3. Mode 2 Structure

Mode 2 configures the timer register as an 8-bit counter(TLx) with automatic reload. Overflow from TLx not only set TFx, but also reload TLx with the content of THx, which is determined by software. The reload leaves THx unchanged. Mode 2 operation is the same for Timer0 and Timer1.

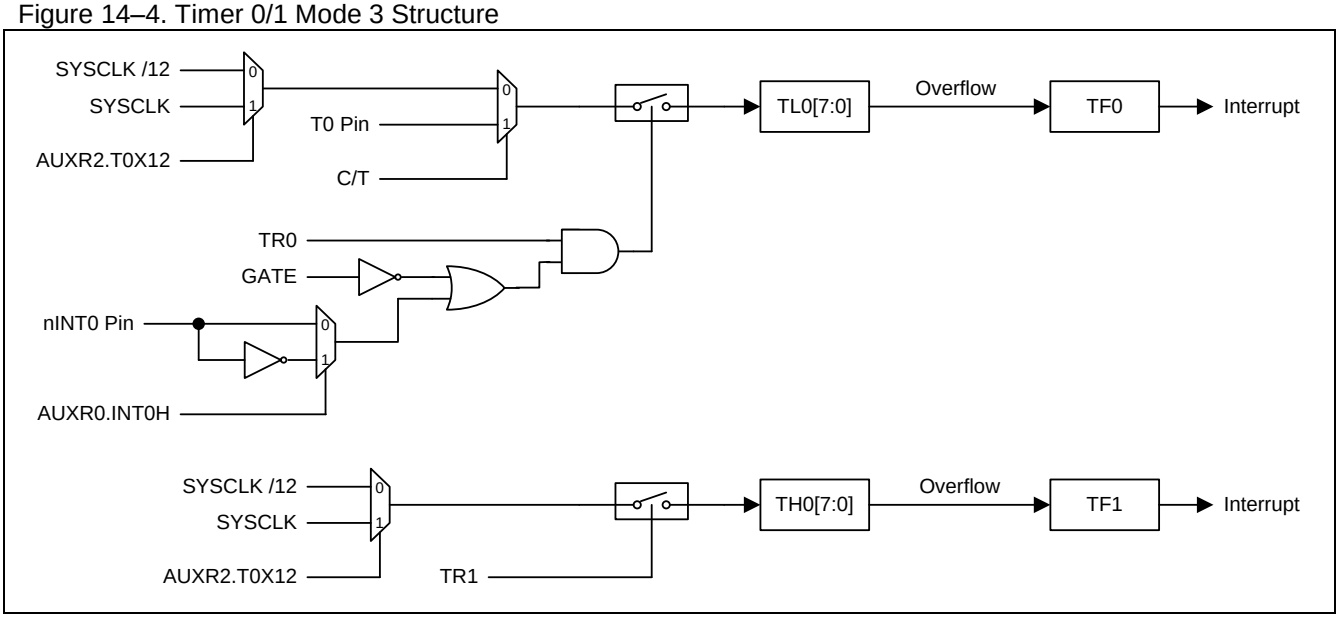
Figure 14–3. Timer 0/1 Mode 2 Structure



### 14.1.4. Mode 3 Structure

Timer1 in Mode3 simply holds its count, the effect is the same as setting TR1 = 1. Timer0 in Mode 3 enables TL0 and TH0 as two separate 8-bit counters. TL0 uses the Timer0 control bits such like C/T, GATE, TR0, INT0 and TF0. TH0 is locked into a timer function (can not be external event counter) and take over the use of TR1, TF1 from Timer1. TH0 now controls the Timer1 interrupt.

Figure 14–4. Timer 0/1 Mode 3 Structure




### 14.1.5. Timer 0/1 Programmable Clock-Out

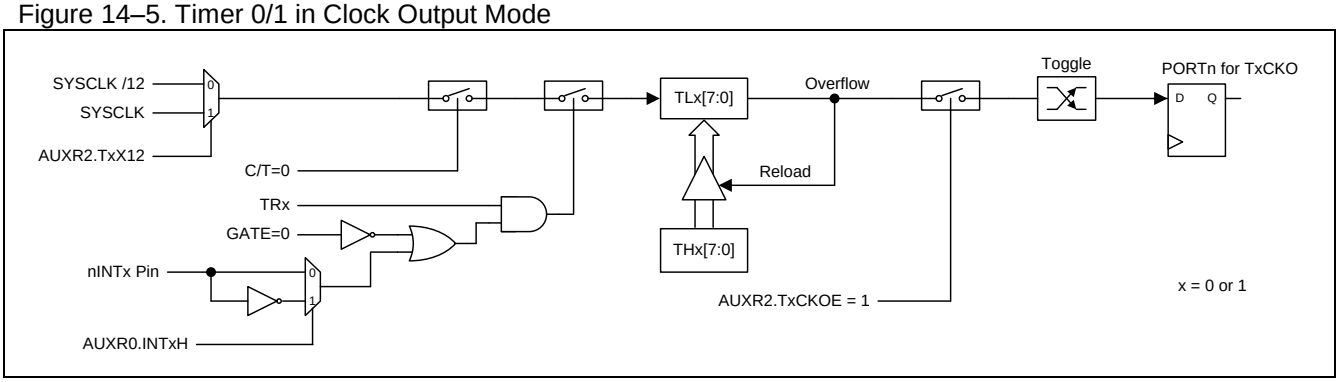
Timer 0 and Timer 1 have a Clock-Out Mode (while C/Tx=0 & TxCKOE=1). In this mode, Timer 0 or Timer 1 operates as 8-bit auto-reload timer for a programmable clock generator with 50% duty-cycle. The generated clocks come out on P3.4 (T0CKO) and P3.5 (T1CKO) individually. The input clock (SYSCLK/12 or SYSCLK) increments the 8-bit timer (TL0 or TL1). The timer repeatedly counts to overflow from a loaded value. Once overflows occur, the contents of (TH0 and TH1) are loaded into (TL0, TL1) for the consecutive counting. The following formula gives the clock-out frequency:

$$\text{T0/T1 Clock-out Frequency} = \frac{\text{SYSCLK Frequency}}{n \times (256 - \text{THx})}$$

; n=24, if TxX12=0
; n=2, if TxX12=1
; x = 0 or 1 & C/T = 0

*Note:*
*(1) Timer 0/1 overflow flag, TF0/1, will be set when Timer 0/1 overflows but not generate interrupt.*
*(2) For SYSCLK=12MHz & TxX12=0, Timer 0/1 has a programmable output frequency range from 1.95KHz to 500KHz.*
*(3) For SYSCLK=12MHz & TxX12=1, Timer 0/k1 has a programmable output frequency range from 23.43KHz to 6MHz.*

Figure 14–5. Timer 0/1 in Clock Output Mode




**How to Program Timer 0/1 in Clock-out Mode**

- Select T0X12/T1X12 bit in AUXR2 register to decide the Timer 0/1 clock source.
- Set T0CKOE/T1CKOE bit in AUXR2 register.
- Clear C/T bit in TMOD register.
- Determine the 8-bit reload value from the formula and enter it in the TH0/TH1 register.
- Enter the same reload value as the initial value in the TL0/TL1 register.
- Set TR0/TR1 bit in TCON register to start the Timer 0/1.

In the Clock-Out mode, Timer 0/1 rollovers will not generate an interrupt. This is similar to when Timer 1 is used as a baud-rate generator. It is possible to use Timer 1 as a baud rate generator and a clock generator simultaneously. Note, however, that the baud-rate and the clock-out frequency depend on the same overflow rate of Timer 1.

### 14.1.6. Timer0/1 Register

***TCON: Timer/Counter Control Register***

SFR Page = Normal
SFR Address = 0x88 RESET = 0000-0000

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| TF1 | TR1 | TF0 | TR0 | IE1 | IT1 | IE0 | IT0 |
| **R/W** | **R/W** | **R/W** | **R/W** | **R/W** | **R/W** | **R/W** | **R/W** |

Bit 7: TF1, Timer 1 overflow flag.
0: Cleared by hardware when the processor vectors to the interrupt routine, or cleared by software.
1: Set by hardware on Timer/Counter 1 overflow, or set by software.

Bit 6: TR1, Timer 1 Run control bit.
0: Cleared by software to turn Timer/Counter 1 off.
1: Set by software to turn Timer/Counter 1 on.

Bit 5: TF0, Timer 0 overflow flag.
0: Cleared by hardware when the processor vectors to the interrupt routine, or cleared by software.
1: Set by hardware on Timer/Counter 0 overflow, or set by software.

Bit 4: TR0, Timer 0 Run control bit.
0: Cleared by software to turn Timer/Counter 0 off.
1: Set by software to turn Timer/Counter 0 on.

***TMOD: Timer/Counter Mode Control Register***

SFR Page = Normal
SFR Address = 0x89 RESET = 0000-0000

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| GATE | C/T | M1 | M0 | GATE | C/T | M1 | M0 |
| **R/W** | **R/W** | **R/W** | **R/W** | **R/W** | **R/W** | **R/W** | **R/W** |
| \|←---------- Timer1 ----------→\| | | | | \|←---------- Timer0 ----------→\| | | |

Bit 7/3: Gate, Gating control for Timer1/0.
0: Disable gating control for Timer1/0.
1: Enable gating control for Timer1/0. When set, Timer1/0 or Counter1/0 is enabled only when /INT1 or /INT0 pin is high and TR1 or TR0 control bit is set.

Bit 6/2: C/T, Timer for Counter function selector.
0: Clear for Timer operation, input from internal system clock.
1: Set for Counter operation, input form T1 input pin.

Bit 5~4/1~0: Operating mode selection.

| **M1** | **M0** | **Operating Mode** |
|---|---|---|
| 0 | 0 | 13-bit timer/counter for Timer0 and Timer1 |
| 0 | 1 | 16-bit timer/counter for Timer0 and Timer1 |
| 1 | 0 | 8-bit timer/counter with automatic reload for Timer0 and Timer1 |
| 1 | 1 (Timer0) | TL0 is 8-bit timer/counter, TH0 is locked into 8-bit timer |
| 1 | 1 (Timer1) | Timer/Counter1 Stopped |

***TL0: Timer Low 0 Register***

SFR Page = Normal

SFR Address = 0x8A RESET = 0000-0000

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| TL0.7 | TL0.6 | TL0.5 | TL0.4 | TL0.3 | TL0.2 | TL0.1 | TL0.0 |
| **R/W** | **R/W** | **R/W** | **R/W** | **R/W** | **R/W** | **R/W** | **R/W** |

***TH0: Timer High 0 Register***

SFR Page = Normal

SFR Address = 0x8C RESET = 0000-0000

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| TH0.7 | TH0.6 | TH0.5 | TH0.4 | TH0.3 | TH0.2 | TH0.1 | TH0.0 |
| **R/W** | **R/W** | **R/W** | **R/W** | **R/W** | **R/W** | **R/W** | **R/W** |

***TL1: Timer Low 1 Register***

SFR Page = Normal

SFR Address = 0x8B RESET = 0000-0000

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| TL1.7 | TL1.6 | TL1.5 | TL1.4 | TL1.3 | TL1.2 | TL1.1 | TL1.0 |
| **R/W** | **R/W** | **R/W** | **R/W** | **R/W** | **R/W** | **R/W** | **R/W** |

***TH1: Timer High 1 Register***

SFR Page = Normal

SFR Address = 0x8D RESET = 0000-0000

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| TH1.7 | TH1.6 | TH1.5 | TH1.4 | TH1.3 | TH1.2 | TH1.1 | TH1.0 |
| **R/W** | **R/W** | **R/W** | **R/W** | **R/W** | **R/W** | **R/W** | **R/W** |

***AUXR2: Auxiliary Register 2***

SFR Page = Normal

SFR Address = 0xA3 RESET = 0000-0000

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| URXR | BTI | URM0X6 | SMOD2 | **T1X12** | **T0X12** | T1CKOE | T0CKOE |
| **R/W** | **R/W** | **R/W** | **R/W** | **R/W** | **R/W** | **R/W** | **R/W** |

Bit 3: T1X12, Timer 1 clock source selector while C/T=0.
0: Clear to select SYSCLK/12.
1: Set to select SYSCLK as the clock source.

Bit 2: T0X12, Timer 0 clock source selector while C/T=0.
0: Clear to select SYSCLK/12.
1: Set to select SYSCLK as the clock source.

Bit 1: T1CKOE, Timer 1 Clock Output Enable.
0: Disable Timer 1 clock output.
1: Enable Timer 1 clock output on P3.5.

Bit 0: T0CKOE, Timer 0 Clock Output Enable.
0: Disable Timer 0 clock output.
1: Enable Timer 0 clock output on P3.4.

### 14.1.7. Timer0/1 Sample Code

*(1). Required Function: IDLE mode with T0 wake-up frequency 320Hz, SYSCLK = ILRCO*

Assembly Code Example:

```
    ORG     0000Bh
time0_isr:
    to do…
    RETI

main:                                       ; (unsigned short value)
    //Switch Sysclk to ILRCO
    MOV     IFADRL,#(CKCON2)                ; Index Page-P address to CKCON2
    CALL    _page_p_sfr_read                ; Read CKCON2 data

    ANL     IFD,#~(OSCS1 | OSCS0)           ; Switch OSCin source to ILRCO
    ORL     IFD,#(OSCS1)
    CALL    _page_p_sfr_write               ; Write data to CKCON2

    ANL     IFD,#~(XTALE | IHRCOE)          ; Disable XTAL and IHRCO
    CALL    _page_p_sfr_write               ; Write data to CKCON2

    MOV     IFADRL,#(DCON0)                 ; Index Page-P address to DCON0
    CALL    _page_p_sfr_read                ; Read DCON0 data

    ANL     IFD,#~(HSE)                     ; Disable HSE when SYSCLK ≤ 6MHz for power saving
    CALL    _page_p_sfr_write               ; Write data to DCON0

    ANL     CKCON0,#(AFS)                   ; Select SCKS[2:0] = 0 = OSCin/1

    ORL     AUXR2,#T0X12                    ; Select SYSCLK/1 for Timer 0 clock input
    ANL     AUXR0,#~T0XL                    ;

    MOV     TH0,#(256-100)                  ; Set Timer 0 overflow rate = SYSCLK x 100
    MOV     TL0,#(256-100)                  ;
    ANL     TMOD,#(0F0h|T0M1)               ; Set Timer 0 to Mode 2
    ORL     TMOD,#T0M1                      ;
    CLR     TF0                             ; Clear Timer 0 Flag

    ORL     IP0L,#PT0L                      ; Select Timer 0 interrupt priority
    ORL     IP0H,#PT0H                      ;

    SETB    ET0                             ; Enable Timer 0 interrupt
    SETB    EA                              ; Enable global interrupt

    SETB    TR0                             ; Start Timer 0 running

    ORL     PCON0,#IDL                      ; Set MCU into IDLE mode
```

C Code Example:

```
void time0_isr(void) interrupt 1
{
    To do…
}

void main(void)
{
    IFADRL = CKCON2;                        // Index Page-P address to CKCON2
    page_p_sfr_read();                      // Read CKCON2 data.

    IFD = ~(OSCS1 | OSCS0);                 // Switch OSCin source to ILRCO
    IFD |= OSCS1;
    page_p_sfr_write();                     // Write data to CKCON2
```

```
    IFD &= ~(XTALE | IHRCOE);                 // Disable XTAL and IHRCO
    page_p_sfr_write();                       // Write data to CKCON2

    IFADRL = DCON0;                           // Index Page-P address to DCON0
    page_p_sfr_read();                        // Read DCON0 data

    IFD &= ~HSE;                              // Disable HSE when SYSCLK ≤ 6MHz for power saving
    page_p_sfr_write();                       // Write data to DCON0

    CKCON0 &= AFS;                            // Select SCKS[2:0] = 0 = OSCin/1

    AUXR2 |= T0X12;                           // Select SYSCLK/1 for Timer 0 clock input
    AUXR0 &= ~T0XL;

    TH0 = TL0 = (256-100);                    // Set Timer 0 overflow rate = SYSCLK x 100
    TMOD &= 0xF0;                             // Set Timer 0 to Mode 2
    TMOD |= T0M1;
    TF0 = 0;                                  // Clear Timer 0 Flag

    IP0L |= PT0L;                             // Select Timer 0 interrupt priority
    IP0H |= PT0H;

    ET0 = 1;                                  // Enable Timer 0 interrupt
    EA = 1;                                   // Enable global interrupt

    TR0 = 1;                                  // Start Timer 0 running

    PCON0=IDL;                                // Set MCU into IDLE mode
}
```

*(2). Required Function: Set Timer 0 clock output by SYSCLK/48 input*

Assembly Code Example:

```
    CLR     TR0                               ;

    ANL     P3M0,#0EFh                        ; Set P3.4(T0CKO) to push-pull output
    ORL     P3M1,#010h                        ;
    ORL     AUXR2,#T0CKOE                     ; Enable T0CKO

    ANL     AUXR2,#~T0X12                     ; Select SYSCLK/48 for Timer 0 clock input
    ORL     AUXR0,#T0XL                       ;

    MOV     TH0,#0FFh                         ;
    MOV     TL0,#0FFh                         ;

    ANL     TMOD,#0F0h                        ; Set Timer 0 to Mode 2
    ORL     TMOD,#T0M1                        ;

    SETB    TR0                               ; Start Timer 0 running
```

C Code Example:

```
    TR0 = 0;

    P3M0 &= 0xEF;                             // Set P3.4(T0CKO) to push-pull output
    P3M1 |= 0x10;
    AUXR2 |= T0CKOE;                          // Enable T0CKO

    AUXR2 &= ~T0X12;                          // Select SYSCLK/48 for Timer 0 clock input
    AUXR0 |= T0XL;

    TH0 = TL0 = 0xFF;

    TMOD &= 0xF0;                             // Set Timer 0 to Mode 2
    TMOD |= T0M1;

    TR0 = 1;                                  // Start Timer 0 running
```

*(3). Required Function: Set Timer 1 clock output by SYSCLK input*

Assembly Code Example:

```
ORL     P3M1,#020h                  ; Set P3.5(T1CKO) to push-pull output
ANL     P3M0,#0DFh                  ;

ORL     AUXR2,#(T1X12|T1CKOE)       ; Select SYSCLK for Timer 1 clock input
                                    ; Enable T1CKO

MOV     TH1,#0FFh                   ;
MOV     TL1,#0FFh                   ;

ANL     TMOD,#00Fh                  ; Set Timer 1 to Mode 2
ORL     TMOD,#T1M1                  ;

SETB    TR1                         ; Start Timer 1 running
```

C Code Example:

```
P3M1 |= 0x20;                       // Set P3.5(T1CKO) to push-pull output
P3M0 &= 0xDF;

AUXR2 |= (T1X12|T1CKOE);            // Select SYSCLK for Timer 1 clock input
                                    // Enable T1CKO

TH1 = TL1 = 0xFF;

TMOD &= 0x0F;                       // Set Timer 1 to Mode 2
TMOD |= T1M1;

TR1 = 1;                            // Start Timer 1 running
```

# 15. Serial Port (UART)

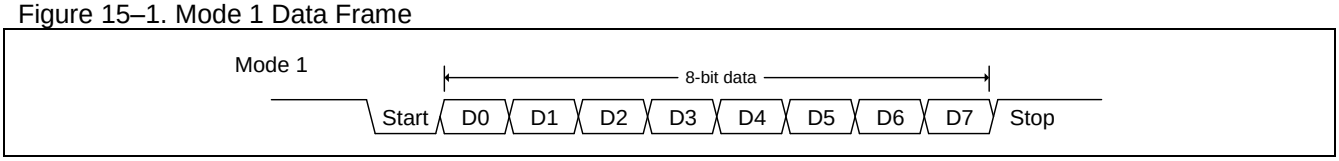
The serial port of **MG86FE/L104** support full-duplex transmission, meaning it can transmit and receive simultaneously. It is also receive-buffered, meaning it can commence reception of a second byte before a previously received byte has been read from the register. However, if the first byte still hasn't been read by the time reception of the second byte is complete, one of the bytes will be lost. The serial port receive and transmit registers are both accessed at special function register SBUF. Writing to SBUF loads the transmit register, and reading from SBUF accesses a physically separate receive register.

The serial port can operate in 4 modes: Mode 0 provides *synchronous* communication while Modes 1, 2, and 3 provide *asynchronous* communication. The asynchronous communication operates as a full-duplex Universal Asynchronous Receiver and Transmitter (UART), which can transmit and receive simultaneously and at different baud rates.

**Mode 0:** 8 data bits (LSB first) are transmitted or received through RXD(P3.0). TXD(P3.1) always outputs the shift clock. The baud rate can be selected to 1/12 or 1/2 the system clock frequency by URM0X6 setting in AUXR2 register.

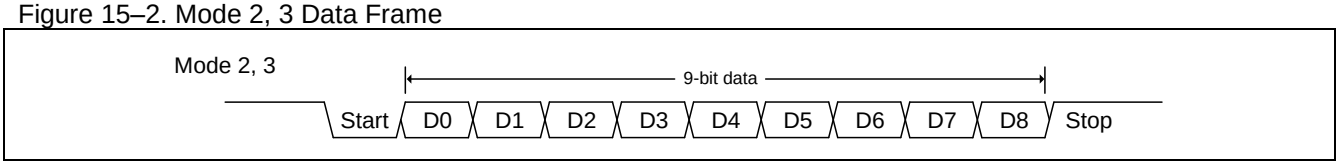
**Mode 1:** 10 bits are transmitted through TXD or received through RXD. The frame data includes a start bit (0), 8 data bits (LSB first), and a stop bit (1), as shown in Figure 15–1. On receive, the stop bit would be loaded into RB8 in SCON register. The baud rate is variable.

Figure 15–1. Mode 1 Data Frame




**Mode 2:** 11 bits are transmitted through TXD or received through RXD. The frame data includes a start bit (0), 8 data bits (LSB first), a programmable 9th data bit, and a stop bit (1), as shown in Figure 15–2. On Transmit, the 9th data bit comes from TB8 in SCON register can be assigned the value of 0 or 1. On receive, the 9th data bit would be loaded into RB8 in SCON register, while the stop bit is ignored. The baud rate can be configured to 1/32 or 1/64 the system clock frequency.

Figure 15–2. Mode 2, 3 Data Frame




**Mode 3:** Mode 3 is the same as Mode 2 except the baud rate is variable.

In all four modes, transmission is initiated by any instruction that uses SBUF as a destination register. In Mode 0, reception is initiated by the condition RI=0 and REN=1. In the other modes, reception is initiated by the incoming start bit with 1-to-0 transition if REN=1.

In addition to the standard operation, the UART can perform framing error detection by looking for missing stop bits, and automatic address recognition.

## 15.1. Serial Port Mode 0

Serial data enters and exits through RXD. TXD outputs the shift clock. 8 bits are transmitted/received: 8 data bits (LSB first). The shift clock source can be selected to 1/12 or 1/2 the system clock frequency by URM0X6 setting in AUXR2 register. Figure 15–3 shows a simplified functional diagram of the serial port in Mode 0.

Transmission is initiated by any instruction that uses SBUF as a destination register. The “write to SBUF” signal triggers the UART engine to start the transmission. The data in the SBUF would be shifted into the RXD(P3.0) pin by each raising edge shift clock on the TXD(P3.1) pin. After eight raising edge of shift clocks passing, TI would be asserted by hardware to indicate the end of transmission. Figure 15–4 shows the transmission waveform in Mode 0.

Reception is initiated by the condition REN=1 and RI=0. At the next instruction cycle, the Serial Port Controller writes the bits 11111110 to the receive shift register, and in the next clock phase activates Receive.

Receive enables Shift Clock which directly comes from RX Clock to the alternate output function of P3.1 pin. When Receive is active, the contents on the RXD(P3.0) pin would be sampled and shifted into shift register by falling edge of shift clock. After eight falling edge of shift clock, RI would be asserted by hardware to indicate the end of reception. Figure 15–5 shows the reception waveform in Mode 0.

Figure 15–3. Serial Port 0 Mode 0

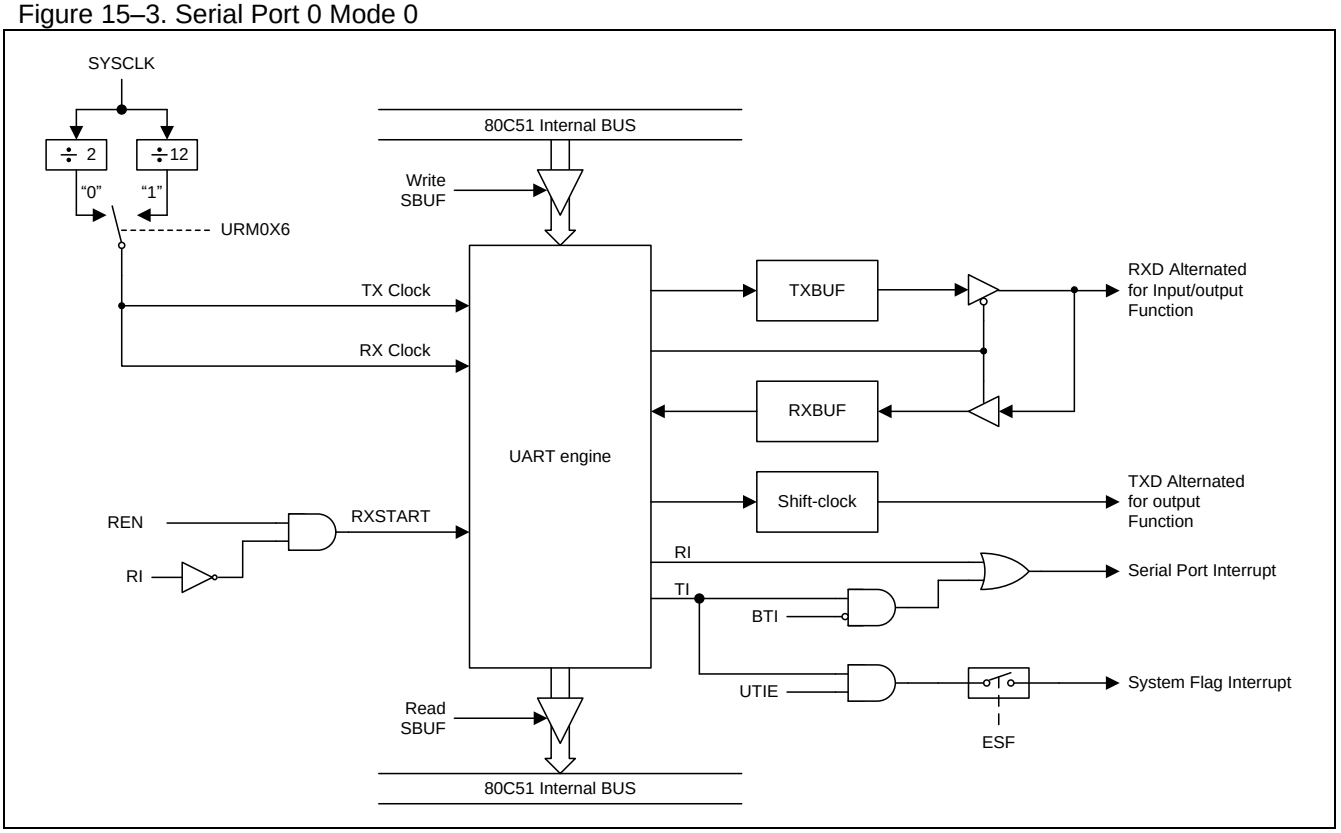


Figure 15–4. Mode 0 Transmission Waveform

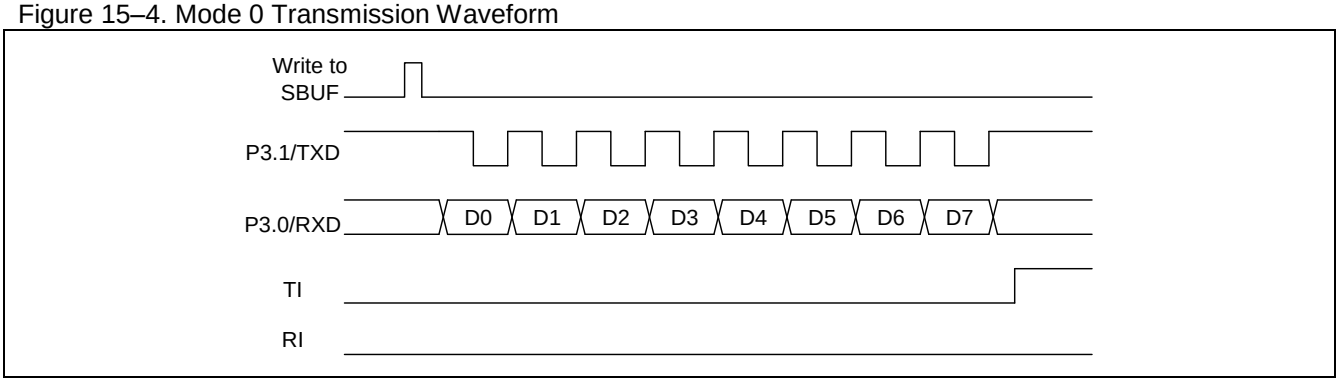
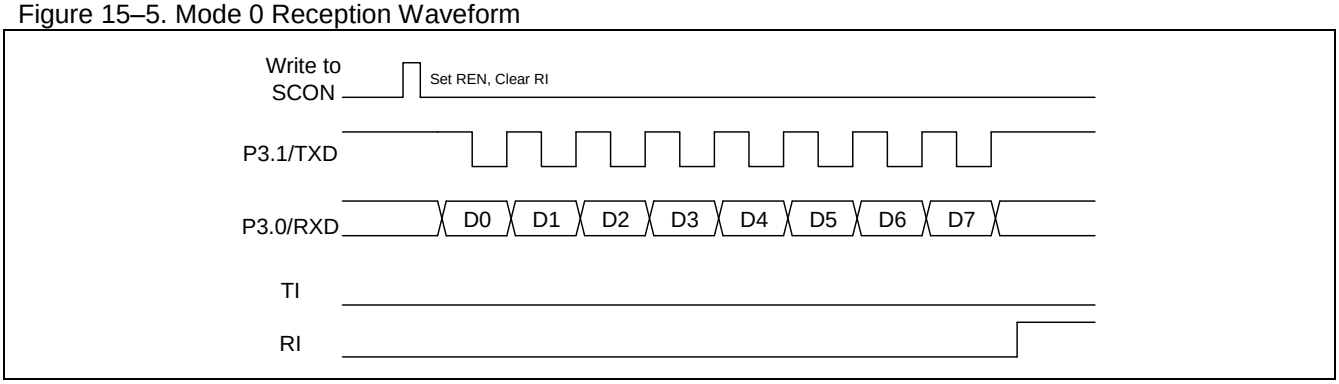



Figure 15–5. Mode 0 Reception Waveform



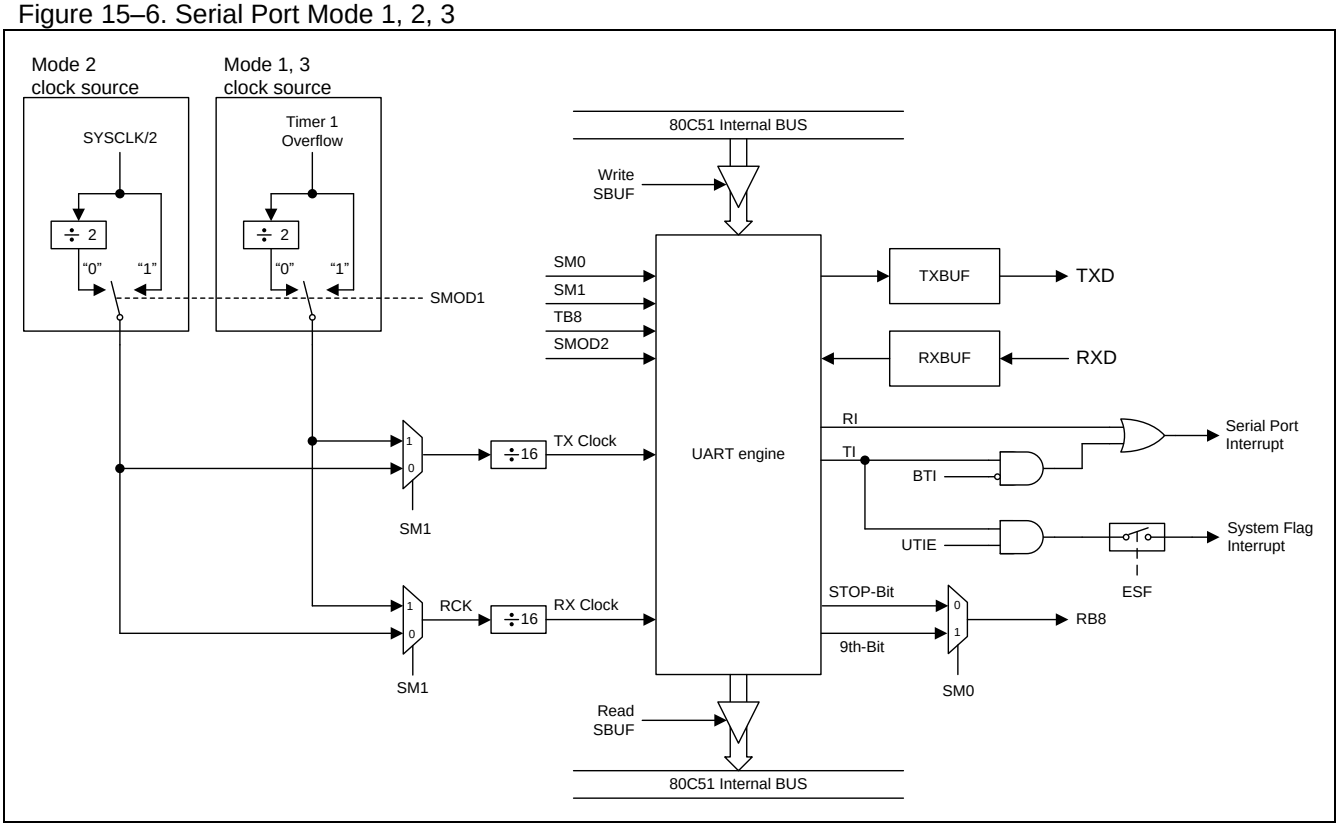
## 15.2. Serial Port Mode 1

10 bits are transmitted through TXD, or received through RXD: a start bit (0), 8 data bits (LSB first), and a stop bit (1). On receive, the stop bit goes into RB8 in SCON. The baud rate is determined by the Timer 1 overflow rate. Figure 15–1 shows the data frame in Mode 1 and Figure 15–6 shows a simplified functional diagram of the serial port in Mode 1.

Transmission is initiated by any instruction that uses SBUF as a destination register. The “write to SBUF” signal requests the UART engine to start the transmission. After receiving a transmission request, the UART engine would start the transmission at the raising edge of TX Clock. The data in the SBUF would be serial output on the TXD pin with the data frame as shown in Figure 15–1 and data width depend on TX Clock. After the end of 8th data transmission, TI would be asserted by hardware to indicate the end of data transmission.

Reception is initiated when Serial Port Controller detected 1-to-0 transition at RXD sampled by RCK. The data on the RXD pin would be sampled by Bit Detector in Serial Port Controller. After the end of STOP-bit reception, RI would be asserted by hardware to indicate the end of data reception and load STOP-bit into RB8 in SCON register.

Figure 15–6. Serial Port Mode 1, 2, 3



## 15.3. Serial Port Mode 2 and Mode 3

11 bits are transmitted through TXD, or received through RXD: a start bit (0), 8 data bits (LSB first), a programmable 9th data bit, and a stop bit (1). On transmit, the 9th data bit (TB8) can be assigned the value of 0 or 1. On receive, the 9th data bit goes into RB8 in SCON. The baud rate is programmable to select one of 1/16, 1/32 or 1/64 the system clock frequency in Mode 2. Mode 3 may have a variable baud rate generated from Timer 1 or Timer 2.

Figure 15–2 shows the data frame in Mode 2 and Mode 3. Figure 15–6 shows a functional diagram of the serial port in Mode 2 and Mode 3. The receive portion is exactly the same as in Mode 1. The transmit portion differs from Mode 1 only in the 9th bit of the transmit shift register.

The “write to SBUF” signal requests the Serial Port Controller to load TB8 into the 9th bit position of the transmit shift register and starts the transmission. After receiving a transmission request, the UART engine would start the transmission at the raising edge of TX Clock. The data in the SBUF would be serial output on the TXD pin with the data frame as shown in Figure 15–2 and data width depend on TX Clock. After the end of 9th data transmission, TI would be asserted by hardware to indicate the end of data transmission.

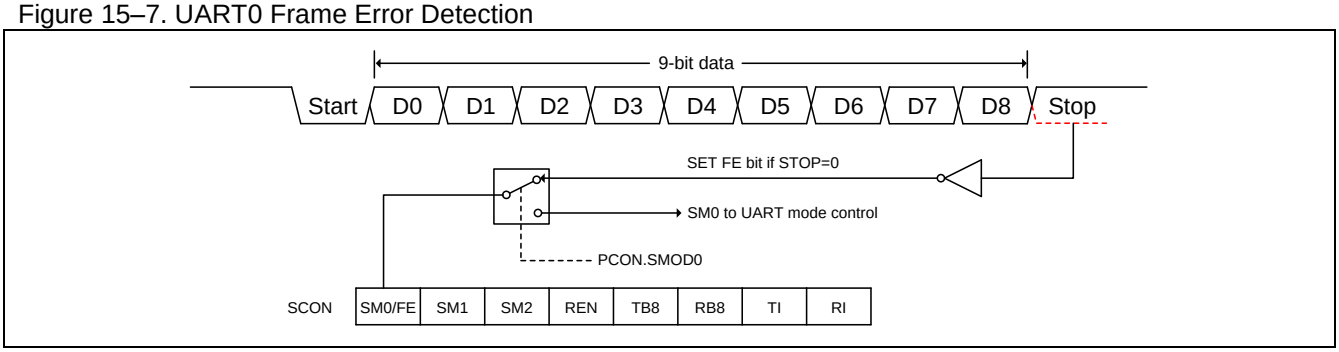
Reception is initiated when the UART engine detected 1-to-0 transition at RXD sampled by RCK. The data on the RXD pin would be sampled by Bit Detector in UART engine. After the end of 9th data bit reception, RI would be asserted by hardware to indicate the end of data reception and load the 9th data bit into RB8 in SCON register.

In all four modes, transmission is initiated by any instruction that use SBUF as a destination register. Reception is initiated in mode 0 by the condition RI = 0 and REN = 1. Reception is initiated in the other modes by the incoming start bit with 1-to-0 transition if REN=1.

## 15.4. Frame Error Detection

When used for framing error detection, the UART looks for missing stop bits in the communication. A missing stop bit will set the FE bit in the SCON register. The FE bit shares the SCON.7 bit with SM0 and the function of SCON.7 is determined by SMOD0 bit (PCON.6). If SMOD0 is set then SCON.7 functions as FE. SCON.7 functions as SM0 when SMOD0 is cleared. When SCON.7 functions as FE, it can only be cleared by firmware. Refer to Figure 15–7.

Figure 15–7. UART0 Frame Error Detection



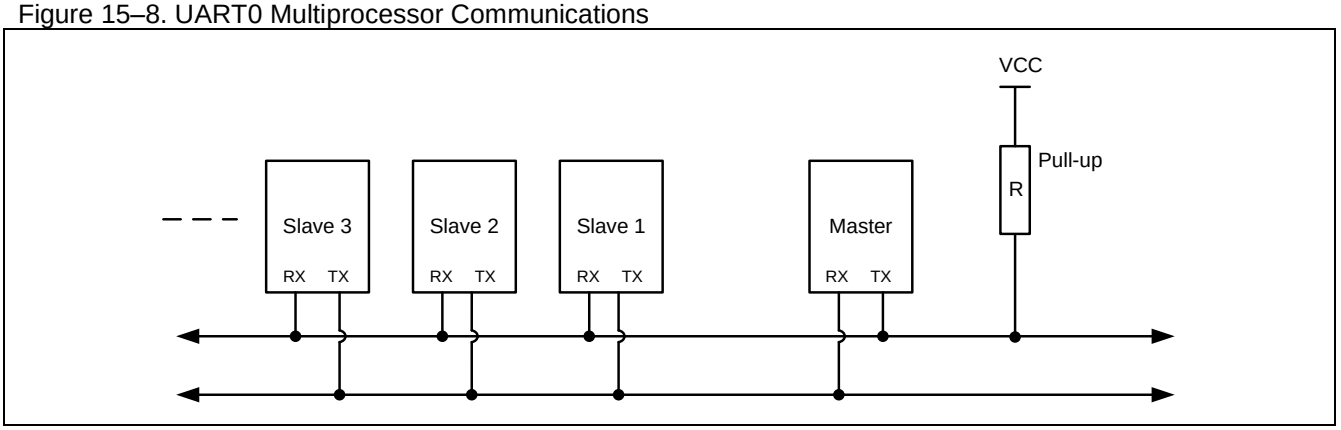
## 15.5. Multiprocessor Communications

Modes 2 and 3 have a special provision for multiprocessor communications as shown in Figure 15–8. In these two modes, 9 data bits are received. The 9th bit goes into RB8. Then comes a stop bit. The port can be programmed such that when the stop bit is received, the serial port interrupt will be activated only if RB8=1. This feature is enabled by setting bit SM2 (in SCON register). A way to use this feature in multiprocessor systems is as follows:

When the master processor wants to transmit a block of data to one of several slaves, it first sends out an address byte which identifies the target slave. An address byte differs from a data byte in that the 9th bit is 1 in an address byte and 0 in a data byte. With SM2=1, no slave will be interrupted by a data byte. An address byte, however, will interrupt all slaves, so that each slave can examine the received byte and check if it is being addressed. The addressed slave will clear its SM2 bit and prepare to receive the data bytes that will be coming. The slaves that weren't being addressed leave their SM2 set and go on about their business, ignoring the coming data bytes.

SM2 has no effect in Mode 0, and in Mode 1 can be used to check the validity of the stop bit. In a Mode 1 reception, if SM2=1, the receive interrupt will not be activated unless a valid stop bit is received.

Figure 15–8. UART0 Multiprocessor Communications




## 15.6. Automatic Address Recognition

Automatic Address Recognition is a feature which allows the UART to recognize certain addresses in the serial bit stream by using hardware to make the comparisons. This feature saves a great deal of firmware overhead by eliminating the need for the firmware to examine every serial address which passes by the serial port. This feature is enabled by setting the SM2 bit in SCON.

In the 9 bit UART modes, mode 2 and mode 3, the Receive Interrupt flag (RI) will be automatically set when the received byte contains either the "Given" address or the "Broadcast" address. The 9-bit mode requires that the 9th information bit is a 1 to indicate that the received information is an address and not data. Automatic address recognition is shown in Figure 15–9. The 8 bit mode is called Mode 1. In this mode the RI flag will be set if SM2 is enabled and the information received has a valid stop bit following the 8 address bits and the information is either a Given or Broadcast address. Mode 0 is the Shift Register mode and SM2 is ignored.

Using the Automatic Address Recognition feature allows a master to selectively communicate with one or more slaves by invoking the Given slave address or addresses. All of the slaves may be contacted by using the Broadcast address. Two special Function Registers are used to define the slave's address, SADDR, and the address mask, SADEN.

SADEN is used to define which bits in the SADDR are to be used and which bits are "don't care". The SADEN mask can be logically ANDed with the SADDR to create the "Given" address which the master will use for addressing each of the slaves. Use of the Given address allows multiple slaves to be recognized while excluding others.

The following examples will help to show the versatility of this scheme:

| Slave 0 | Slave 1 |
|---|---|
| SADDR = 1100 0000 | SADDR = 1100 0000 |
| SADEN = 1111 1101 | SADEN = 1111 1110 |
| Given = 1100 00X0 | Given = 1100 000X |

In the above example SADDR is the same and the SADEN data is used to differentiate between the two slaves. Slave 0 requires a 0 in bit 0 and it ignores bit 1. Slave 1 requires a 0 in bit 1 and bit 0 is ignored. A unique address for Slave 0 would be 1100 0010 since slave 1 requires a 0 in bit 1. A unique address for slave 1 would be 1100 0001 since a 1 in bit 0 will exclude slave 0. Both slaves can be selected at the same time by an address which has bit 0 = 0 (for slave 0) and bit 1 = 0 (for slave 1). Thus, both could be addressed with 1100 0000.

In a more complex system the following could be used to select slaves 1 and 2 while excluding slave 0:

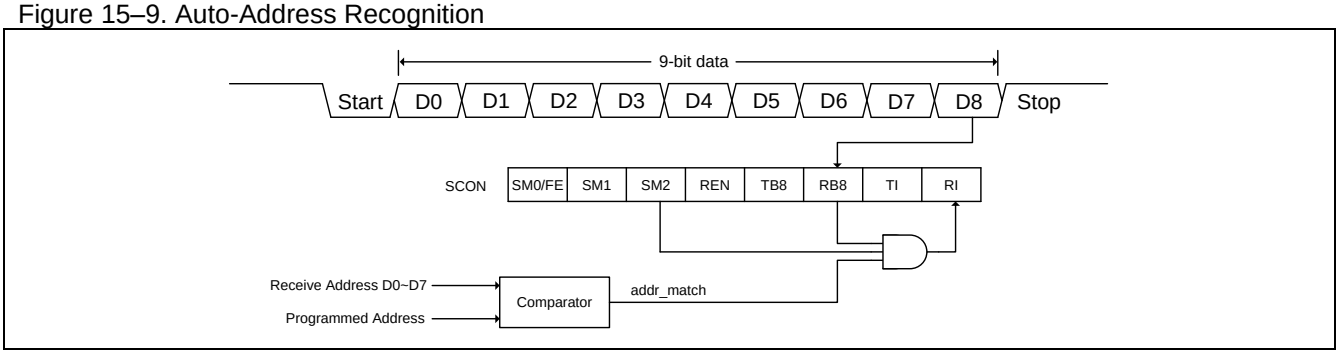
| Slave 0 | Slave 1 | Slave 2 |
|---|---|---|
| SADDR = 1100 0000 | SADDR = 1110 0000 | SADDR = 1110 0000 |
| SADEN = 1111 1001 | SADEN = 1111 1010 | SADEN = 1111 1100 |
| Given = 1100 0XX0 | Given = 1110 0X0X | Given = 1110 00XX |

In the above example the differentiation among the 3 slaves is in the lower 3 address bits. Slave 0 requires that bit 0 = 0 and it can be uniquely addressed by 1110 0110. Slave 1 requires that bit 1 = 0 and it can be uniquely addressed by 1110 0101. Slave 2 requires that bit 2 = 0 and its unique address is 1110 0011. To select Slaves 0 and 1 and exclude Slave 2 use address 1110 0100, since it is necessary to make bit 2 = 1 to exclude slave 2.

The Broadcast Address for each slave is created by taking the logical OR of SADDR and SADEN. Zeros in this result are treated as don't-cares. In most cases, interpreting the don't-cares as ones, the broadcast address will be FF hexadecimal.

Upon reset SADDR (SFR address 0xA9) and SADEN (SFR address 0xB9) are loaded with 0s. This produces a given address of all "don't cares" as well as a Broadcast address of all "don't cares". This effectively disables the Automatic Addressing mode and allows the micro-controller to use standard 80C51 type UART drivers which do not make use of this feature.

Figure 15–9. Auto-Address Recognition




*Note: (1) After address matching(addr_match=1), Clear SM2 to receive data bytes*
*(2) After all data bytes have been received, Set SM2 to wait for next address.*

## 15.7. Baud Rate Setting

Bits T1X12, URM0X6 and SMOD2 in AUXR2 register provide a new option for the baud rate setting, as listed below.

### 15.7.1. Baud Rate in Mode 0

$$\text{Mode 0 Baud Rate} = \frac{F_{SYSCLK}}{n}$$ ; n=12, if URM0X6=0 ; n=2, if URM0X6=1

*Note:*

*If URM0X6=0, the baud rate formula is as same as standard 8051.*

### 15.7.2. Baud Rate in Mode 2

$$\text{Mode 2 Baud Rate} = \frac{2^{SMOD1} \times 2^{(SMOD2 \times 2)}}{64} \times (F_{SYSCLK})$$

*Note*:

*If SMOD2=0, the baud rate formula is as same as standard 8051. If SMOD2=1, there is an enhanced function for baud rate setting. Table 15–1 defines the Baud Rate setting with SMOD2 factor in Mode 2 baud rate generator.*

Table 15–1. SMOD2 application criteria in Mode 2

| SMOD2 | SMOD1 | Baud Rate | Note | Recommended Max. Receive Error (%) |
|---|---|---|---|---|
| 0 | 0 | Default Baud Rate | Standard function | ± 3% |
| 0 | 1 | Double Baud Rate | Standard function | ± 3% |
| 1 | 0 | Double Baud Rate **X2** | Enhanced function | ± 2% |
| 1 | 1 | Double Baud Rate **X4** | Enhanced function | ± 1% |

### 15.7.3. Baud Rate in Mode 1 & 3

**Using Timer 1 as the Baud Rate Generator**

$$\text{Mode 1, 3 Baud Rate} = \frac{2^{SMOD1} \times 2^{(SMOD2 \times 2)}}{32} \times \frac{F_{SYSCLK}}{12 \times (256 - TH1)}$$ ; T1X12=0

$$\text{or} = \frac{2^{SMOD1} \times 2^{(SMOD2 \times 2)}}{32} \times \frac{F_{SYSCLK}}{1 \times (256 - TH1)}$$ ; T1X12=1

*Note*:

*If SMOD2=0, T1X12=0, the baud rate formula is as same as standard 8051. If SMOD2=1, there is an enhanced function for baud rate setting. Table 15–2 defines the Baud Rate setting with SMOD2 factor in Timer 1 baud rate generator.*

Table 15–2. SMOD2 application criteria in Mode 1 & 3 using Timer 1

| SMOD2 | SMOD1 | Baud Rate | Note | Recommended Max. Receive Error (%) |
|---|---|---|---|---|
| 0 | 0 | Default Baud Rate | Standard function | ± 3% |
| 0 | 1 | Double Baud Rate | Standard function | ± 3% |
| 1 | 0 | Double Baud Rate **X2** | Enhanced function | ± 2% |
| 1 | 1 | Double Baud Rate **X4** | Enhanced function | ± 1% |

Table 15-3 ~ Table 15-10 list various commonly used baud rates and how they can be obtained from Timer 1 in its 8-Bit Auto-Reload Mode.

Table 15–3. Timer 1 Generated Commonly Used Baud Rates @ $F_{SYSCLK}$=11.0592MHz

| Baud Rate | **TH1**, the Reload Value | | | | | |
|---|---|---|---|---|---|---|
| | **T1X12=0 & SMOD2=0** | | | **T1X12=1 & SMOD2=0** | | |
| | SMOD1=0 | SMOD1=1 | Error | SMOD1=0 | SMOD1=1 | Error |
| 1200 | 232 | 208 | 0.0% | -- | -- | -- |
| 2400 | 244 | 232 | 0.0% | 112 | -- | 0.0% |
| 4800 | 250 | 244 | 0.0% | 184 | 112 | 0.0% |
| 9600 | 253 | 250 | 0.0% | 220 | 184 | 0.0% |
| 14400 | 254 | 252 | 0.0% | 232 | 208 | 0.0% |
| 19200 | -- | 253 | 0.0% | 238 | 220 | 0.0% |
| 28800 | 255 | 254 | 0.0% | 244 | 232 | 0.0% |
| 38400 | -- | -- | -- | 247 | 238 | 0.0% |
| 57600 | -- | 255 | 0.0% | 250 | 244 | 0.0% |
| 115200 | -- | -- | -- | 253 | 250 | 0.0% |
| 230400 | -- | -- | -- | -- | 253 | 0.0% |

Table 15–4. Timer 1 Generated High Baud Rates @ $F_{SYSCLK}$ =11.0592MHz

| Baud Rate | **TH1**, the Reload Value | | | | | |
|---|---|---|---|---|---|---|
| | **T1X12=0 & SMOD2=1** | | | **T1X12=1 & SMOD2=1** | | |
| | SMOD1=0 | SMOD1=1 | Error | SMOD1=0 | SMOD1=1 | Error |
| 230.4K | -- | 255 | 0.0% | 250 | 244 | 0.0% |
| 460.8K | -- | -- | -- | 253 | 250 | 0.0% |
| 691.2K | -- | -- | -- | 254 | 252 | 0.0% |
| 921.6K | -- | -- | -- | -- | 253 | 0.0% |
| 1.3824M | -- | -- | -- | 255 | 254 | 0.0% |
| 2.7648M | -- | -- | -- | -- | 255 | 0.0% |

Table 15–5. Timer 1 Generated Commonly Used Baud Rates @ $F_{SYSCLK}$=22.1184MHz

| Baud Rate | **TH1**, the Reload Value | | | | | |
|---|---|---|---|---|---|---|
| | **T1X12=0 & SMOD2=0** | | | **T1X12=1 & SMOD2=0** | | |
| | SMOD1=0 | SMOD1=1 | Error | SMOD1=0 | SMOD1=1 | Error |
| 1200 | 208 | 160 | 0.0% | -- | -- | -- |
| 2400 | 232 | 208 | 0.0% | -- | -- | 0.0% |
| 4800 | 244 | 232 | 0.0% | 112 | -- | 0.0% |
| 9600 | 250 | 244 | 0.0% | 184 | 112 | 0.0% |
| 14400 | 252 | 248 | 0.0% | 208 | 160 | 0.0% |
| 19200 | 253 | 250 | 0.0% | 220 | 184 | 0.0% |
| 28800 | 254 | 252 | 0.0% | 232 | 208 | 0.0% |
| 38400 | -- | 253 | 0.0% | 238 | 220 | 0.0% |
| 57600 | 255 | 254 | 0.0% | 244 | 232 | 0.0% |
| 115200 | -- | 255 | 0.0% | 250 | 244 | 0.0% |
| 230400 | -- | -- | -- | 253 | 250 | 0.0% |
| 460800 | -- | -- | -- | -- | 253 | 0.0% |

Table 15–6. Timer 1 Generated High Baud Rates @ $F_{SYSCLK}$=22.1184MHz

| Baud Rate | **TH1**, the Reload Value | | | | | |
|---|---|---|---|---|---|---|
| | **T1X12=0 & SMOD2=1** | | | **T1X12=1 & SMOD2=1** | | |
| | SMOD1=0 | SMOD1=1 | Error | SMOD1=0 | SMOD1=1 | Error |
| 460.8K | -- | 255 | 0.0% | 250 | 244 | 0.0% |
| 691.2K | -- | -- | -- | 252 | 248 | 0.0% |
| 921.6K | -- | -- | -- | 253 | 250 | 0.0% |
| 1.3824M | -- | -- | -- | 254 | 252 | 0.0% |
| 1.8432M | | | | -- | 253 | 0.0% |
| 2.7648M | -- | -- | -- | 255 | 254 | 0.0% |
| 5.5296M | -- | -- | -- | -- | 255 | 0.0% |

Table 15–7. Timer 1 Generated Commonly Used Baud Rates @ $F_{SYSCLK}$=12.0MHz

| Baud Rate | **TH1**, the Reload Value | | | | | |
|---|---|---|---|---|---|---|
| | **T1X12=0 & SMOD2=0** | | | **T1X12=1 & SMOD2=0** | | |
| | SMOD1=0 | SMOD1=1 | Error | SMOD1=0 | SMOD1=1 | Error |
| 1200 | 230 | 204 | 0.16% | -- | -- | -- |
| 2400 | 243 | 230 | 0.16% | 100 | -- | 0.16% |
| 4800 | -- | 243 | 0.16% | 178 | 100 | 0.16% |
| 9600 | -- | -- | -- | 217 | 178 | 0.16% |
| 14400 | -- | -- | -- | 230 | 204 | 0.16% |
| 19200 | -- | -- | -- | -- | 217 | 0.16% |
| 28800 | -- | -- | -- | 243 | 230 | 0.16% |
| 38400 | -- | -- | -- | 246 | 236 | 2.34% |
| 57600 | -- | -- | -- | -- | 243 | 0.16% |
| 115200 | -- | -- | -- | -- | -- | -- |

Table 15–8. Timer 1 Generated High Baud Rates @ $F_{SYSCLK}$=12.0MHz

| Baud Rate | **TH1**, the Reload Value | | | | | |
|---|---|---|---|---|---|---|
| | **T1X12=0 & SMOD2=1** | | | **T1X12=1 & SMOD2=1** | | |
| | SMOD1=0 | SMOD1=1 | Error | SMOD1=0 | SMOD1=1 | Error |
| 115.2K | -- | -- | -- | 243 | 230 | 0.16% |
| 230.4K | -- | -- | -- | -- | 243 | 0.16% |
| 460.8K | -- | -- | -- | -- | -- | -- |

Table 15–9. Timer 1 Generated Commonly Used Baud Rates @ $F_{SYSCLK}$=24.0MHz

| Baud Rate | **TH1**, the Reload Value | | | | | |
|---|---|---|---|---|---|---|
| | **T1X12=0 & SMOD2=0** | | | **T1X12=1 & SMOD2=0** | | |
| | SMOD1=0 | SMOD1=1 | Error | SMOD1=0 | SMOD1=1 | Error |
| 1200 | 204 | 152 | 0.16% | -- | -- | -- |
| 2400 | 230 | 204 | 0.16% | -- | -- | -- |
| 4800 | 243 | 230 | 0.16% | 100 | -- | 0.16% |
| 9600 | -- | 243 | 0.16% | 178 | 100 | 0.16% |
| 14400 | -- | -- | -- | 204 | 152 | 0.16% |
| 19200 | -- | -- | -- | 217 | 178 | 0.16% |
| 28800 | -- | -- | -- | 230 | 204 | 0.16% |
| 38400 | -- | -- | -- | -- | 217 | 0.16% |
| 57600 | -- | -- | -- | 243 | 230 | 0.16% |
| 115200 | -- | -- | -- | -- | 243 | 0.16% |

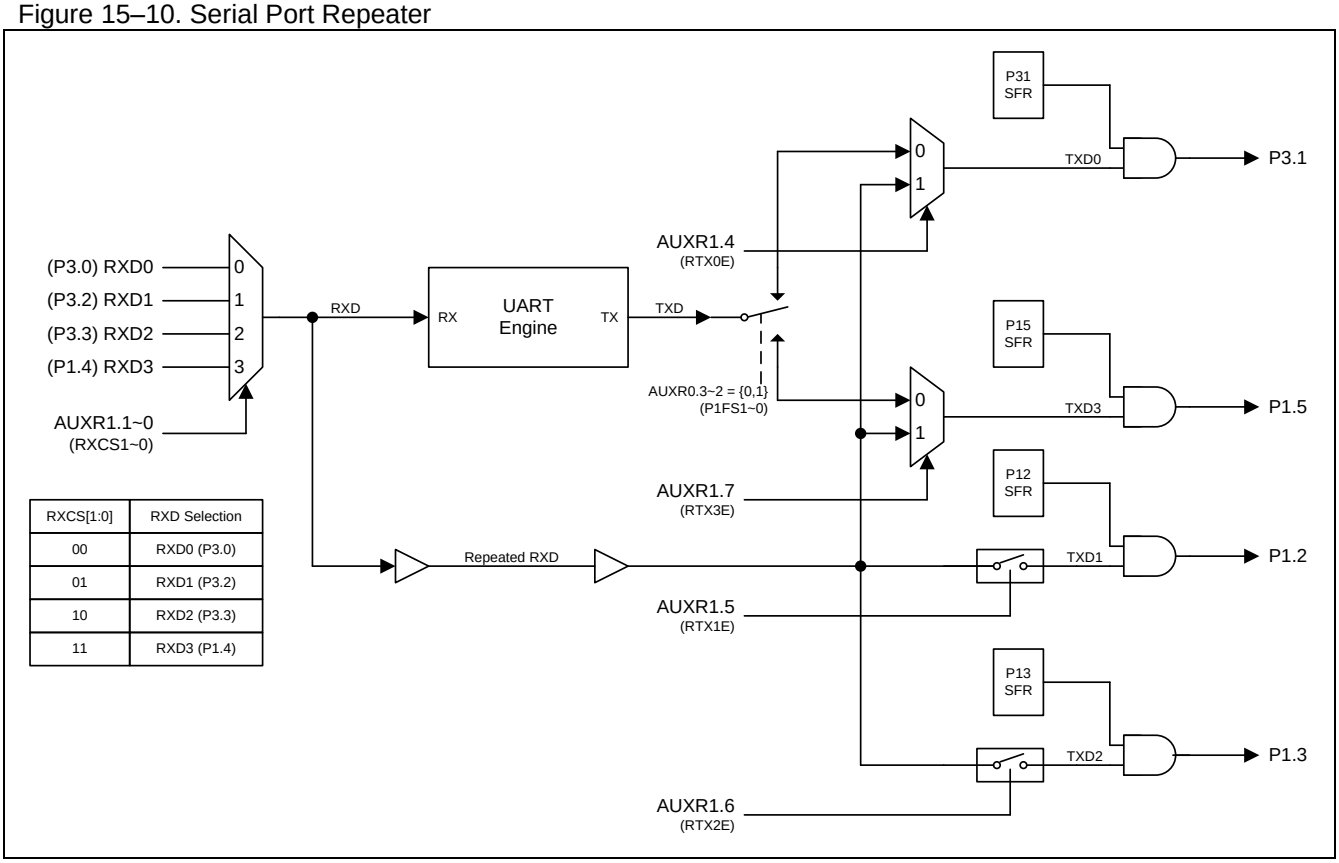
Table 15–10. Timer 1 Generated High Baud Rates @ $F_{SYSCLK}$=24.0MHz

| Baud Rate | **TH1**, the Reload Value | | | | | |
|---|---|---|---|---|---|---|
| | **T1X12=0 & SMOD2=1** | | | **T1X12=1 & SMOD2=1** | | |
| | SMOD1=0 | SMOD1=1 | Error | SMOD1=0 | SMOD1=1 | Error |
| 230.4K | -- | -- | -- | 243 | 230 | 0.16% |
| 460.8K | -- | -- | -- | -- | 243 | 0.16% |
| 691.2K | -- | -- | -- | -- | -- | -- |
| 921.6K | -- | -- | -- | -- | -- | -- |

## 15.8. Serial Port Repeater Mode

Figure 15–10 shows the serial port repeater in MG86FE/L104. Software can configure the data flow path from RXD to TXD. It can select one of four RXD input into UART engine and directly route to 4 TXD outputs concurrently. The repeater function only supports serial port mode 1, 2 and 3.

Figure 15–10. Serial Port Repeater




| RXCS[1:0] | RXD Selection |
|---|---|
| 00 | RXD0 (P3.0) |
| 01 | RXD1 (P3.2) |
| 10 | RXD2 (P3.3) |
| 11 | RXD3 (P1.4) |

## 15.9. Serial Port Register

All the four operation modes of the serial port are the same as those of the standard 8051 except the baud rate setting. Three registers, PCON, AUXR and AUXR2, are related to the baud rate setting:

***SCON: Serial port Control Register***

SFR Page = Normal

SFR Address = 0x98 RESET = 0000-0000

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| SM0/FE | SM1 | SM2 | REN | TB8 | RB8 | TI | RI |
| **R/W** | **R/W** | **R/W** | **R/W** | **R/W** | **R/W** | **R/W** | **R/W** |

Bit 7: FE, Framing Error bit. The SMOD0 bit must be set to enable access to the FE bit.
0: The FE bit is not cleared by valid frames but should be cleared by software.
1: This bit is set by the receiver when an invalid stop bit is detected.

Bit 7: Serial port mode bit 0, (SMOD0 must = 0 to access bit SM0)
Bit 6: Serial port mode bit 1.

| SM0 | SM1 | Mode | Description | Baud Rate |
|---|---|---|---|---|
| 0 | 0 | 0 | shift register | SYSCLK/12 or /2 |
| 0 | 1 | 1 | 8-bit UART | variable |
| 1 | 0 | 2 | 9-bit UART | SYSCLK/64, /32, /16 or /8 |
| 1 | 1 | 3 | 9-bit UART | variable |

Bit 5: Serial port mode bit 2.
0: Disable SM2 function.
1: Enable the automatic address recognition feature in Modes 2 and 3. If SM2=1, RI will not be set unless the received 9th data bit is 1, indicating an address, and the received byte is a Given or Broadcast address. In mode1, if SM2=1 then RI will not be set unless a valid stop Bit was received, and the received byte is a Given or Broadcast address. In Mode 0, SM2 should be 0.

Bit 4: REN, Enable serial reception.
0: Clear by software to disable reception.
1: Set by software to enable reception.

Bit 3: TB8, The 9th data bit that will be transmitted in Modes 2 and 3. Set or clear by software as desired.

Bit 2: RB8, In Modes 2 and 3, the $9^{th}$ data bit that was received. In Mode 1, if SM2 = 0, RB8 is the stop bit that was received. In Mode 0, RB8 is not used.

Bit 1: TI. Transmit interrupt flag.
0: Must be cleared by software.
1: Set by hardware at the end of the $8^{th}$ bit time in Mode 0, or at the beginning of the stop bit in the other modes, in any serial transmission.

Bit 0: RI. Receive interrupt flag.
0: Must be cleared by software.
1: Set by hardware at the end of the $8^{th}$ bit time in Mode 0, or halfway through the stop bit time in the other modes, in any serial reception (except see SM2).

***SBUF: Serial Buffer Register***

SFR Page = Normal

SFR Address = 0x99 RESET = XXXX-XXXX

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| SBUF.7 | SBUF.6 | SBUF.5 | SBUF.4 | SBUF.3 | SBUF.2 | SBUF.1 | SBUF.0 |
| **R/W** | **R/W** | **R/W** | **R/W** | **R/W** | **R/W** | **R/W** | **R/W** |

Bit 7~0: It is used as the buffer register in transmission and reception.

***SADDR: Slave Address Register***

SFR Page = Normal

SFR Address = 0xA9 RESET = 0000-0000

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| | | | | | | | |
| **R/W** | **R/W** | **R/W** | **R/W** | **R/W** | **R/W** | **R/W** | **R/W** |

***SADEN: Slave Address Mask Register***

SFR Page = Normal

SFR Address = 0xB9 RESET = 0000-0000

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| | | | | | | | |
| **R/W** | **R/W** | **R/W** | **R/W** | **R/W** | **R/W** | **R/W** | **R/W** |

SADDR register is combined with SADEN register to form Given/Broadcast Address for automatic address recognition. In fact, SADEN functions as the “mask” register for SADDR register. The following is the example for it.

SADDR = 1100 0000
SADEN = 1111 1101
Given = 1100 00x0 ⟶ The Given slave address will be checked except bit 1 is treated as “don’t care”

The Broadcast Address for each slave is created by taking the logical OR of SADDR and SADEN. Zero in this result is considered as “don’t care”. Upon reset, SADDR and SADEN are loaded with all 0s. This produces a Given Address of all “don’t care” and a Broadcast Address of all “don’t care”. This disables the automatic address detection feature.

***PCON0: Power Control Register 0***

SFR Page = Normal & Page P

SFR Address = 0x87 POR = 00X1-0000, RESET = 00X0-0000

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| SMOD1 | SMOD0 | GF | POF | GF1 | GF0 | PD | IDL |
| **R/W** | **R/W** | **R/W** | **R/W** | **R/W** | **R/W** | **R/W** | **R/W** |

Bit 7: SMOD1, double Baud rate control bit.
0: Disable double Baud rate of the UART.
1: Enable double Baud rate of the UART in mode 1, 2, or 3.

Bit 6: SMOD0, Frame Error select.
0: SCON.7 is SM0 function.
1: SCON.7 is FE function. Note that FE will be set after a frame error regardless of the state of SMOD0.

***AUXR1: Auxiliary Register 1***

SFR Page = Normal

SFR Address = 0xA2 RESET = 0000-0000

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| RTX3E | RTX2E | RTX1E | RTX0E | -- | -- | RXCS1 | RXCS0 |
| **R/W** | **R/W** | **R/W** | **R/W** | **W** | **W** | **R/W** | **R/W** |

Bit 7: RTX3E, RXD Repeat to TXD3 enabled.
0: Disable RTX3E function on TXD3 (P1.5).
1: Enable the RXD repeat mechanism on TXD3 (P1.5).

Bit 6: RTX2E, RXD Repeat to TXD2 enabled.
0: Disable RTX2E function on TXD2 (P1.3).
1: Enable the RXD repeat mechanism on TXD3 (P1.3).

Bit 5: RTX1E, RXD Repeat to TXD1 enabled.
0: Disable RTX1E function on TXD1 (P1.2).
1: Enable the RXD repeat mechanism on TXD1 (P1.2).

Bit 4: RTX0E, RXD Repeat to TXD0 enabled.
0: Disable RTX0E function on TXD0 (P3.1).
1: Enable the RXD repeat mechanism on TXD0 (P3.1).

Bit 3~2: Reserved. Software must write “0” on these bits when AUXR1 is written.

Bit 1~0: RXD channel selection.

| RXCS[1:0] | RXD channel selection |
|---|---|
| 00 | P3.0 |
| 01 | P3.2 |
| 10 | P3.3 |
| 11 | P1.4 |

***AUXR2: Auxiliary Register 2***

SFR Page = Normal

SFR Address = 0xA3 RESET = 0000-0000

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| URXR | BTI | URM0X6 | SMOD2 | T1X12 | T0X12 | T1CKOE | T0CKOE |
| **R/W** | **R/W** | **R/W** | **R/W** | **R/W** | **R/W** | **R/W** | **R/W** |

Bit 7: URXR, Serial Port test mode. Software must write “0” on this bit when AUXR2 is written.

Bit 6: BTI, Block TI in Serial Port Interrupt.
0: Retain the TI to be a source of Serial Port Interrupt.
1: Block TI to be a source of Serial Port Interrupt.

Bit 5: URM0X6, Serial Port mode 0 baud rate selector.
0: Clear to select SYSCLK/12 as the baud rate for UART Mode 0.
1: Set to select SYSCLK/2 as the baud rate for UART Mode 0.

Bit 4: SMOD2, extra X2/X4 baud rate selector.
0: Disable extra X2/X4 baud rate.
1: Enable extra X2/X4 baud rate.

Bit 3: T1X12, Timer 1 clock source selector while C/T=0.
0: Clear to select SYSCLK/12.
1: Set to select SYSCLK as the clock source.

***SFIE: System Flag Interrupt Enable Register***

SFR Page = Normal

SFR Address = 0x8E RESET = 0xxx-0x00

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| UTIE | -- | -- | -- | KBIFIE | -- | BOF0IE | WDTFIE |
| **R/W** | **W** | **W** | **W** | **R/W** | **W** | **R/W** | **R/W** |

Bit 7: UTIE, UART TI Enabled in system flag interrupt.
0: Disable the interrupt vector sharing for TI in system flag interrupt.
1: Set TI flag will share the interrupt vector with system flag interrupt.

## 15.10. Serial Port Sample Code

*(1). Required Function: IDLE mode with RI wake-up capability*

Assembly Code Example:

```
      ORG       00023h
uart_ri_idle_isr:
      JB        RI,RI_ISR                       ;
      JB        TI,TI_ISR                       ;
      RETI                                      ;

RI_ISR:
;     Process
      CLR       RI                              ;
      RETI                                      ;

TI_ISR:
;     Process
      CLR       TI                              ;
      RETI                                      ;

main:
      CLR       TI                              ;
      CLR       RI                              ;
      SETB      SM1                             ;
      SETB      REN                             ; 8bit Mode2, Receive Enable

      MOV       IP0L,#PSL                       ; Select S0 interrupt priority
      MOV       IP0H,#PSH                       ;

      SETB      ES                              ; Enable S0 interrupt
      SETB      EA                              ; Enable global interrupt

      ORL       PCON0,#IDL;                     ; Set MCU into IDLE mode
```

C Code Example:

```
void uart_ri_idle_isr(void) interrupt 4
{     if(RI)
      {
            RI=0;
            // to do ...
      }

      if(TI)
      {
            TI=0;
            // to do ...
      }
}

void main(void)
{
      TI = RI = 0;
      SM1 = REN = 1;                            // 8bit Mode2, Receive Enable

      IP0L = PSL;                               // Select S0 interrupt priority
      IP0H = PSH;                               //

      ES = 1;                                   // Enable S0 interrupt
      EA = 1;                                   // Enable global interrupt

      PCON |= IDL;                              // Set MCU into IDLE mode
}
```

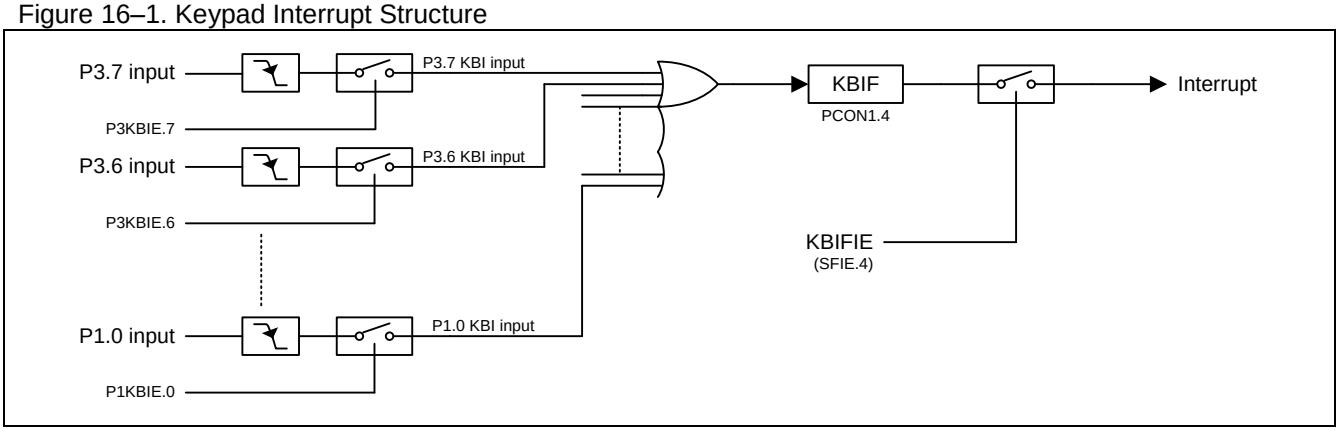
# 16. Keypad Interrupt (KBI)

The Keypad Interrupt function is intended primarily to allow a single interrupt to be generated when enabled port pin is a falling edge occurred. This function can be used keypad recognition. Figure 16–1 shows the structure of the keypad interrupt function.

There are two SFRs used for this function. Each bit in P1KBIE and P3KBIE controls the associated port pin to enable or disable the KBI function at falling edge occurred. Any recognized KBI event will cause the hardware to set the interrupt flag KBIF and generate an interrupt if it has been enabled. Not necessary to enable the KBI interrupt, enabled KBI port pin can wakeup CPU from idle mode (falling edge) and power-down mode (low level).

## 16.1. Keypad Interrupt Structure

Figure 16–1. Keypad Interrupt Structure




## 16.2. Keypad Interrupt Register

***P1KBIE: Port 1 KBI Enable Control Register***

SFR Page = Normal
SFR Address = 0xD7 RESET = 0000-0000

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| P17KBIE | P16KBIE | P15KBIE | P14KBIE | P13KBIE | P12KBIE | P11KBIE | P10KBIE |
| **R/W** | **R/W** | **R/W** | **R/W** | **R/W** | **R/W** | **R/W** | **R/W** |

Bit 7~0: Keypad Input function enable bit for each Port 1 pins.
0: Disable associated port pin for keypad input function.
1: Enable associated port pin for keypad input function.

***P3KBIE: Port 3 KBI Enable Control Register***

SFR Page = Normal
SFR Address = 0xD6 RESET = 0000-0000

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| P37KBIE | P36KBIE | P35KBIE | P34KBIE | **P41KBIE** | **P40KBIE** | P31KBIE | P30KBIE |
| **R/W** | **R/W** | **R/W** | **R/W** | **R/W** | **R/W** | **R/W** | **R/W** |

Bit 7~0: Keypad Input function enable bit for P3.7 ~ P3.4, P4.1, P4.0, P3.1 and P3.0 pins.
0: Disable associated port pin for keypad input function.
1: Enable associated port pin for keypad input function.

***PCON1: Power Control Register 1***

SFR Page = Normal & Page P

SFR Address = 0x97 POR = 0010-0X00

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| SWRF | EXRF | -- | -- | **KBIF** | -- | BOF0 | WDTF |
| **R/W** | **R/W** | **W** | **W** | **R/W** | **W** | **R/W** | **R/W** |

Bit 3: KBIF, Keypad Interrupt Flag.

0: This bit must be cleared by software writing **"1"** on it. Software writing ":0" is no operation.

1: This bit is only set by falling edge on enabled KBI port pin. Writing **"1"** on this bit will clear KBIF. In power down mode, this bit is set by low level on enabled KBI port pin.

## 16.3. Keypad Interrupt Sample Code

*(1). Required Function: Implement a KBI function on P1.3~P1.0*

Assembly Code Example:

```
      ORG       0003Bh
SystemFlag_ISR:                                 ;
      PUSH      ACC                             ;

      MOV       A,PCON1                         ;
      JNB       ACC.3,Not_KBIF_ISR              ;
      To do……
      ANL       PCON1,#KBIF                     ; Clear KBI flag (write “1”)

Not_KBIF_ISR:
      POP       ACC                             ;
      RETI                                      ;

main:
      ORL       PUCON0,#PU10                    ; Enable P1.3~P1.0 on-chip pull-up resistor
      ANL       P1M0,#0F0h                      ; Set P1.3~P1.0 to Open-Drain Output
      ORL       P1,#00Fh                        ; Set P1.3~P1.0 to Input mode

      ORL       EIP1L,#PSFL                     ; Select SystemFlag interrupt priority
      ORL       EIP1H,#PSFH                     ;

      MOV       P1KBIE,#00Fh                    ; Enable P1.3~P1.0 KBI function
;     MOV       P3KBIE,#0xF3                    ; Enable P3 KBI function
;     MOV       P3KBIE,#0x0C                    ; Enable P4.1~0 KBI function

      ANL       PCON1,#KBIF                     ; Clear KBI flag (write “1”)

      ORL       SFIE,#KBIFIE                    ; Enable KBI interrupt
      ORL       EIE1,#ESF                       ; Enable SystemFlag interrupt
      SETB      EA                              ; Enable global interrupt

      ORL       PCON0,#PD                       ; Set MCU into power-down mode
```

C Code Example:

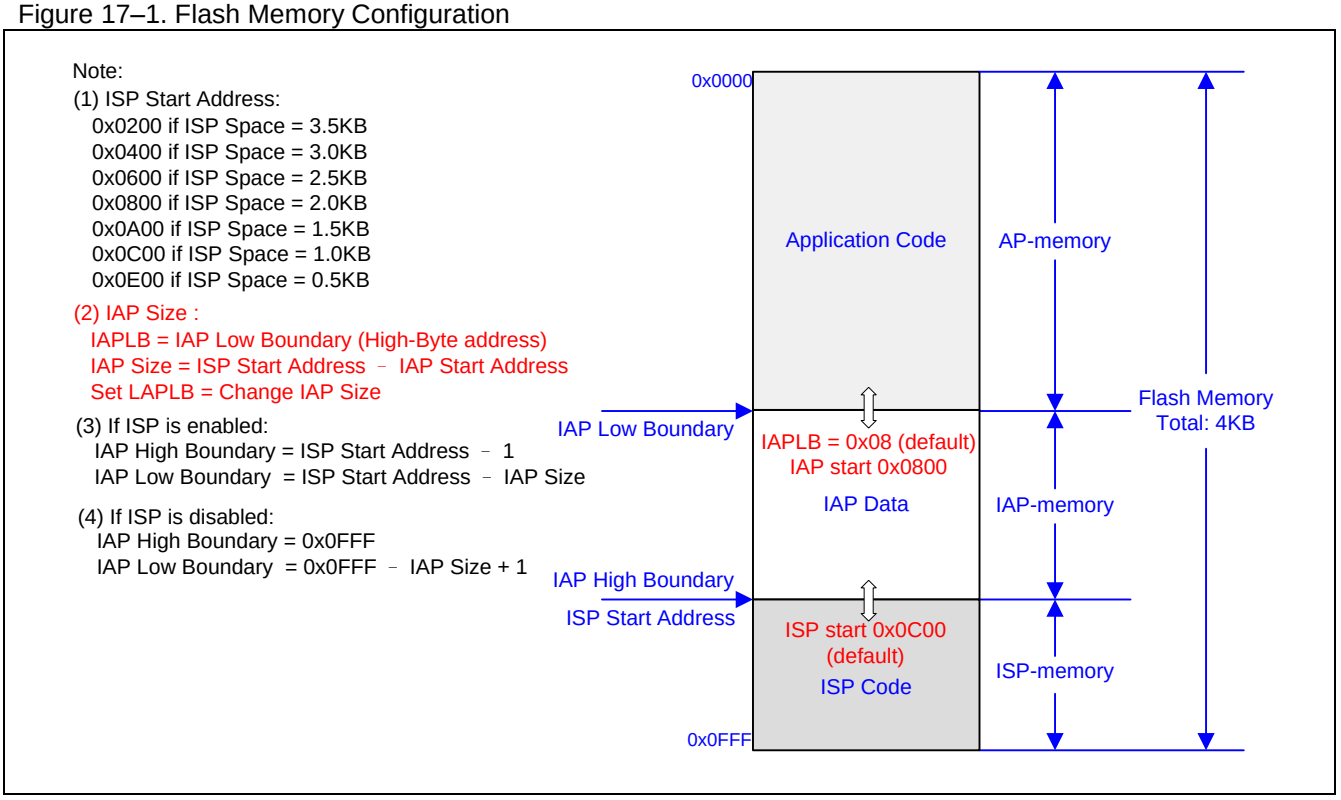
```
void SystemFlag_ISR(void) interrupt 7
{     if(PCON1 & KBIF)
      {
            PCON1 &= KBIF;                      // Clear KBI Flag
      }
}

void main(void)
{
      PUCON0 |= PU10;                           // Enable P1.3~P1.0 on-chip pull-up resistor
      P1M0 &= 0xF0;                             // Set P1.3~P1.0 to Open-Drain Output
      P1 |= 0x0F;                               // Set P1.3~P1.0 to Input mode

      EIP1L |= PSFL;                            // Select SystemFlag interrupt priority
      EIP1H |= PSFH;

      P1KBIE = 0x0F;                            // Enable P1.3~P1.0 KBI function
//    P3KBIE = 0xF3;                            // Enable P3 KBI function
//    P3KBIE = 0x0C;                            // Enable P4.1~0 KBI function
      PCON1&= KBIF;                             // Clear KBI flag (write “1”)

      SFIE |= KBIFIE;                           // Enable KBI interrupt
      EIE1 |= ESF;                              // Enable SystemFlag interrupt
      EA = 1;                                   // Enable global interrupt

      PCON0 |= PD;                              // Set MCU into power-down mode
}
```

# 17. ISP and IAP

## 17.1. Flash Memory Configuration

There are total 4K bytes of Flash Memory in **MG86FE/L104** and Figure 17–1 shows the device flash configuration. The flash can be partitioned into AP-memory, IAP-memory and ISP-memory. AP-memory is used to store user's application program; IAP-memory is used to store the non-volatile application data; and, ISP-memory is used to store the boot loader program for In-System Programming. When MCU is running in ISP region, MCU could modify the AP and IAP memory for software upgraded. If MCU is running in AP region, MCU could only modify the IAP memory for storage data updated.

Figure 17–1. Flash Memory Configuration




*Note:*

*In default, the samples that Megawin shipped had configured the flash memory for 1K ISP, 1K IAP and Lock enabled. The 1K ISP region is inserted Megawin proprietary ISP code to perform In-System-Programming through Megawin 1-Line ISP protocol.*

**MG86FE/L104** does not make use of idle-mode to perform ISP and IAP function. Instead, it freezes CPU running to release the flash memory for ISP or IAP engine operating. Once ISP/IAP operation finished, CPU will be resumed and advanced to the instruction which follows the previous instruction that invokes ISP/AP activity. During ISPIAP operation, interrupt service is also blocked until ISP/IAP finished.

## 17.2. In-System-Programming (ISP)

The flash memory of **MG86FE/L104** can be both programming by the universal Writer/Programmer or the way of In-System Programming (ISP). ISP makes it possible to update the user's application program (in AP-memory) and non-volatile application data (in IAP-memory) without removing the MCU chip from the actual end product. This useful capability makes a wide range of field-update applications possible.

*Note:*

*(1) Before using the ISP feature, the user should configure an ISP-memory space and pre-program the ISP code into the ISP-memory by a universal Writer/Programmer or Megawin proprietary Writer/Programmer.*

*(2) ISP code in the ISP-memory can only program the AP-memory and IAP-memory.*

### 17.2.1. ISP/IAP Register

The following special function registers are related to the access of ISP, IAP and Page-P SFR:

***IFD: ISP/IAP Flash Data Register***

SFR Page = Normal

SFR Address = 0xE2 RESET = 1111-1111

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| | | | | | | | |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

IFD is the data port register for ISP/IAP/Page-P operation. The data in IFD will be written into the desired address in operating ISP/IAP/Page-P write and it is the data window of readout in operating ISP/IAP read.

***IFADRH: ISP/IAP Address for High-byte addressing***

SFR Page = Normal

SFR Address = 0xE3 RESET = 0000-0000

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| | | | | | | | |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

IFADRH is the high-byte address port for all ISP/IAP modes.

***IFADRL: ISP/IAP Address for Low-byte addressing***

SFR Page = Normal

SFR Address = 0xE4 RESET = 0000-0000

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| | | | | | | | |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

IFADRL is the low byte address port for all ISP/IAP/Page-P modes.

***IFMT: ISP/IAP Flash Mode Table***

SFR Page = Normal

SFR Address = 0xE5 RESET = xxxx-x000

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| -- | -- | -- | -- | -- | MS.2 | MS.1 | MS.0 |
| W | W | W | W | W | R/W | R/W | R/W |

Bit 7~4: Reserved. Software must write "0000_0" on these bits when IFMT is written.

Bit 3~0: ISP/IAP/Page-P operating mode selection

| MS[2:0] | Mode |
|---|---|
| 0 0 0 | Standby |
| 0 0 1 | Flash byte read of AP/IAP-memory |
| 0 1 0 | Flash byte program of AP/IAP-memory |
| 0 1 1 | **Reserved** |
| **1 0 0** | **Page P SFR Write** |
| **Others** | **Reserved** |

IFMT is used to select the flash mode for performing numerous ISP/IAP function or to select page P SFR access.

***IAPLB: IAP Low Boundary***

SFR Page = **Page P Only**
SFR Address = 0x03 RESET = 1111-111x

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| IAPLB | | | | | | | 0 |
| **W** | **W** | **W** | **W** | **W** | **W** | **W** | **W** |

Bit 7~0: The IAPLB determines the IAP-memory lower boundary. Since a Flash page has 512 bytes, the IAPLB must be an even number.
To read IAPLB, MCU need to define the IMFT for mode selection on IAPLB Read and set ISPCR.ISPEN. And then write 0x46h & 0xB9h sequentially into SCMD. The IAPLB content is available in IFD. If write IAPLB, MCU will put new IAPLB setting value in IFD firstly. And then select IMFT, enable ISPCR.ISPEN and then set SCMD. The IAPLB content has already finished the updated sequence.

The range of the IAP-memory is determined by IAPLB and the ISP start address as listed below.
*IAP lower boundary = IAPLBx256, and*
*IAP higher boundary = ISP start address – 1.*

For example, if IAPLB=0x08 and ISP start address is 0x0C00, then the IAP-memory range is located at 0x0800 ~ 0x0BFF.

Additional attention point, the IAP low boundary address must not be higher than ISP start address.

***SCMD: Sequential Command Data register***

SFR Page = Normal
SFR Address = 0xE6 RESET = xxxx-xxxx

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| SCMD | | | | | | | |
| **R/W** | **R/W** | **R/W** | **R/W** | **R/W** | **R/W** | **R/W** | **R/W** |

SCMD is the command port for triggering ISP/IAP/Page-P activity. If SCMD is filled with sequential 0x46h, 0xB9h and if ISPCR.7 = 1, ISP/IAP/Page-P activity will be triggered.

***ISPCR: ISP Control Register***

SFR Page = Normal
SFR Address = 0xE7 RESET = 0000-0xxx

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| ISPEN | SWBS | SWRST | CFAIL | -- | -- | -- | -- |
| **R/W** | **R/W** | **R/W** | **R/W** | **W** | **W** | **W** | **W** |

Bit 7: ISPEN, ISP/IAP/Page-P operation enable.
0: Global disable all ISP/IAP/Page-P program/ read function.
1: Enable ISP/IAP/Page-P program/ read function.

Bit 6: SWBS, software boot selection control.
0: Boot from main-memory after reset.
1: Boot from ISP memory after reset.

Bit 5: SWRST, software reset trigger control.
0: No operation
1: Generate software system reset. It will be cleared by hardware automatically.

Bit 4: CFAIL, Command Fail indication for ISP/IAP operation.
0: The last ISP/IAP command has finished successfully.
1: The last ISP/IAP command fails. It could be caused since the access of flash memory was inhibited.

Bit 3~0: Reserved. Software must write “0” on these bits when ISPCR is written.

### 17.2.2. Description for ISP Operation

CPU in **MG86FE/L104** can be vectored into ISP-memory space from two possible ways: When HWBS (in hardware option bits) enabled, **MG86FE/L104** will always boot from the ISP-memory on the specified “ISP start address” since power-on. This way is named as hardware approach. Another way is software approach that allows **MG86FE/L104** execution switched into ISP-memory from AP memory by software setting ISPCR.7 ~ ISPCR.5 to “111” simultaneously.

Once CPU in ISP memory region and ISPEN = 1, accurate values in ISP-related registers should be confirmed by ISP software. Then, sequentially writing 0x46h, 0xB9h into SCMD are used to really trigger memory access operations (flash byte programming and flash byte read).

After ISP operation has been finished, software writes “001” on ISPCR.7 ~ ISPCR.5 which triggers an software RESET and makes CPU reboot into application program memory (AP-memory) on the address 0x0000.

**To do Byte Program**

Step 1: Set MS[2:0]=[0,1,0] in ISPCR register to select Byte Program Mode.
Step 2: Fill byte address in IFADRH & IFADRL registers.
Step 3: Fill data to be programmed in IFD register.
Step 4: Sequentially write 0x46h then 0xB9h to SCMD register to trigger an ISP processing.

**To do Read**

Step 1: Set MS[2:0]=[0,0,1] in ISPCR register to select Read Mode.
Step 2: Fill byte address in IFADRH & IFADRL registers.
Step 3: Sequentially write 0x46h then 0xB9h to SCMD register to trigger an ISP processing.
Step 4: Now, the Flash data is in IFD register.

### 17.2.3. Sample Code for ISP

Figure 17–2. shows a sample code for ISP operation.

Figure 17–2. Sample Code for ISP

```
;*******************************************************************************
; Demo Program for the ISP
;*******************************************************************************
IFD         DATA    0E2h
IFADRH      DATA    0E3h
IFADRL      DATA    0E4h
IFMT        DATA    0E5h
SCMD        DATA    0E6h
ISPCR       DATA    0E7h
;
            MOV     ISPCR,#10000000b    ;ISPCR.7=1, enable ISP

;===============================================================================
; 1. Byte Program Mode
;===============================================================================
            ORL     IFMT,#02h       ;MS[2:0]=[0,1,0], select Byte Program Mode
            ANL     ISPCR,#0FAh     ;
            MOV     IFADRH,??       ;fill byte address in IFADRH & IFADRL
            MOV     IFADRL,??       ;
            MOV     IFD,??          ;fill the data to be programmed in IFD
            MOV     SCMD,#46h       ;trigger ISP processing
            MOV     SCMD,#0B9h      ;
            ;Now in processing...(CPU will halt here until complete)

;===============================================================================
; 2. Verify using Read Mode
;===============================================================================

            ANL     IFMT,#0F9h      ;MS1[2:0]=[0,0,1], select Byte Read Mode
            ORL     IFMT,#01h       ;
            MOV     IFADRH,??       ;fill byte address in IFADRH & IFADRL
            MOV     IFADRL,??       ;
            MOV     SCMD,#46h       ;trigger ISP processing
            MOV     SCMD,#0B9h      ;
            ;Now in processing...(CPU will halt here until complete)
            MOV     A,IFD           ;data will be in IFD
            CJNE    A,wanted,ISP_error ;compare with the wanted value
            ...
ISP_error:
            ...
;
```

## 17.3. In-Application-Programming (IAP)

The device is *In Application Programmable* (IAP), which allows some region in the Flash memory to be used as non-volatile data storage while the application program is running. This useful feature can be applied to the application where the data must be kept after power off. Thus, there is no need to use an external serial EEPROM (such as 93C46, 24C01, .., and so on) for saving the non-volatile data.

In fact, the operating of IAP is the same as that of ISP except the Flash range to be programmed is different. The programmable Flash range for ISP operating is located within the AP and IAP memory, while the range for IAP operating is only located within the configured IAP-memory.

*Note:*

*(1) Before using the IAP feature, the software should specify an IAP-memory space by writing IAPLB in Page-P SFR. The IAP-memory space can be also configured by a universal Writer/Programmer or Megawin proprietary Writer/Programmer which configuration is corresponding to IAPLB initial value.*

*(2) The program code to execute IAP is located in the AP-memory and **just only** program IAP-memory **not** ISP-memory.*

### 17.3.1. IAP-memory Boundary/Range

If ISP-memory is specified, the range of the IAP-memory is determined by IAP and the ISP starts address as listed below.

*IAP high boundary = ISP start address –1.*
*IAP low boundary = ISP start address - IAP.*

If ISP-memory is not specified, the range of the IAP-memory is determined by the following formula.

*IAP high boundary = 0x0FFF.*
*IAP low boundary = 0x0FFF – IAP + 1.*

For example, if ISP-memory is 1K, so that ISP start address is 0x0C00, and IAP-memory is 1K, then the IAP-memory range is located at 0x0800 ~ 0x0BFF. The IAP low boundary in **MG86FE/L104** is defined by IAPLB register which can be modified by software to adjust the IAP size in user's AP program.

### 17.3.2. Update data in IAP-memory

The special function registers are related to ISP/IAP would be shown in Section "17.2.1 ISP/IAP Register".

Except whole chip erase operation, **MG86FE/L104** has no flash erasing operation before any flash byte program. To update "one byte" in the IAP-memory, users can directly program the new datum into the target byte. The following steps show the proper procedure:

Step 1: Program the updated data into target addressing memory. ***(using Byte Program mode of ISP)***.
Step 2: To read the data in the IAP-memory for verification, users can use either the ***"MOVC A,@A+DPTR"*** instruction or the ***Read mode of ISP***.

## 17.4. ISP/IAP Sample Code

*(1). Required Function: General function call for ISP/IAP flash read*

Assembly Code Example:

```
_ixp_read:
ixp_read:
      MOV       ISPCR,#ISPEN                    ; Enable Function
      MOV       IFMT,#MS0                       ; ixp_read=0x01

      MOV       SCMD,#046h                      ;
      MOV       SCMD,#0B9h                      ;

      MOV       IFMT,#000h                      ; Flash_Standby=0x00
      ANL       ISPCR,#~ISPEN                   ; Disable Function

      RET
```

C Code Example:

```
void ixp_read (void)
{
      ISPCR = ISPEN;                            // Enable Function
      IFMT = IxP_Flash_Read;                    // IxP_Read=0x01

      SCMD = 0x46;                              //
      SCMD = 0xB9;                              //

      IFMT = Flash_Standby;                     // Flash_Standby=0x00
      ISPCR &= ~ISPEN;
}
```

*(2). Required Function: General function call for ISP/IAP flash program*

Assembly Code Example:

```
_ixp_program:
ixp_program:
      PUSH      ACC

      PUSH      IFADRH                          ;
      PUSH      IFADRL                          ;
      PUSH      IFD                             ;

      MOV       IFADRL,#WDTCR                   ;
      MOV       IFD,WDTCR                       ;
      ORL       IFD,#CLRW                       ;
      CALL      page_p_sfr_write                ;

      POP       IFD                             ;
      POP       IFADRL                          ;
      POP       IFADRH                          ;

      MOV       ISPCR,#ISPEN                    ; Enable Function
      MOV       IFMT,#MS1                       ; IxP_Write=0x02
      MOV       SCMD,#046h                      ;
      MOV       SCMD,#0B9h                      ;

      PUSH      IFD                             ;
      MOV       A,IFD                           ;
      CPL       A                               ;
      MOV       IFD,A                           ;

      MOV       IFMT,#MS0                       ; IxP_Read=0x01
      MOV       SCMD,#046h                      ;
```

```
        MOV         SCMD,#0B9h                          ;

;       if(reg[2] == IFD)
        POP         ACC                                 ;
        CJNE        A,IFD,ixp_first_progrma_fail        ;
        JMP         ixp_progrma_Pass                    ;

ixp_first_progrma_fail:
;       page_p_sfr_write (WDTCR_P,(WDTCR | CLRW)); //
        MOV         IFD,A                               ;
        PUSH        IFADRH                              ;
        PUSH        IFADRL                              ;
        PUSH        IFD                                 ;

        MOV         IFADRL,#WDTCR                       ;
        MOV         IFD,WDTCR                           ;
        ORL         IFD,#CLRW                           ;
        CALL        page_p_sfr_write                    ;

        POP         IFD                                 ;
        POP         IFADRL                              ;
        POP         IFADRH                              ;

        ANL         ISPCR,#~CFAIL                       ;
        MOV         IFMT,#MS1                           ; ixp_write=0x02
        MOV         SCMD,#046h                          ;
        MOV         SCMD,#0B9h                          ;

        PUSH        IFD                                 ;
        MOV         A,IFD                               ;
        CPL         A                                   ;
        MOV         IFD,A                               ;

        MOV         IFMT,#MS0                           ; IxP_Read=0x01
        MOV         SCMD,#046h                          ;
        MOV         SCMD,#0B9h                          ;

;       if(reg[2] == IFD)
        POP         ACC
        CJNE        A,IFD,ixp_second_progrma_Fail
ixp_progrma_Pass:
;       SETB        ixp_program_state                   ; Pass equ 1
        CLR         ixp_program_state                   ; Pass equ 0

end_ixp_program:
        MOV         IFMT,#000h                          ;
        ANL         ISPCR,#~ISPEN                       ;

        POP         ACC                                 ;
        RET                                             ;

ixp_second_progrma_Fail:
;       CLR         ixp_program_state                   ; Fail equ 0
        SETB        ixp_program_state                   ; Fail equ 1
        JMP         end_ixp_program                     ;
```

C Code Example:

```
bit ixp_program(void)
{   unsigned char reg[3];

        reg[0] = IFADRH;                                //
        reg[1] = IFADRL;                                //
        reg[2] = IFD;

        IFADRL = WDTCR_P;                               //
        IFD = (WDTCR | CLRW);                           //
        page_p_sfr_write ();                            //
```

```
    IFADRH = reg[0];                        //
    IFADRL = reg[1];                        //
    IFD = reg[2];                           //

    ISPCR = ISPEN;                          // Enable Function
    IFMT = IxP_Flash_Program;               // IxP_Write=0x02

    SCMD = 0x46;
    SCMD = 0xB9;

    IFD = ~reg[2];                          //

    IFMT = IxP_Flash_Read;                  // IxP_Read=0x01
    SCMD = 0x46;                            //
    SCMD = 0xB9;                            //

    if(reg[2] == IFD)                       //
      {
        IFMT = Flash_Standby;               // Flash_Standby=0x00
        ISPCR &= ~ISPEN;                    //
        return(Pass);                       //
      }
    else                                    //
    {   IFADRL = WDTCR_P;                   //
        IFD = (WDTCR | CLRW);               //
        page_p_sfr_write ();                //

        IFADRH = reg[0];                    //
        IFADRL = reg[1];                    //
        IFD = reg[2];                       //

        ISPCR &= ~CFAIL;                    //
        IFMT = IxP_Flash_Program;           //IxP_Write=0x02

        SCMD = 0x46;                        //
        SCMD = 0xB9;                        //

        IFD = ~reg[2];                      //

        IFMT = IxP_Flash_Read;              //IxP_Read=0x01
        SCMD = 0x46;                        //
        SCMD = 0xB9;                        //

        IFMT = Flash_Standby;               //Flash_Standby=0x00
        ISPCR &= ~ISPEN;                    //

        if(reg[2] != IFD)                   //
        {
            return(Fail);                   //
        }
                else                        //
        {
            return(Pass);                   //
        }
    }
}
```

*(3). Required Function: ISP Sample code with CFAIL Checked and CLR WDT by hardware approached mode, Write twice if CFAIL true.*

Assembly Code Example:

```
isp_hw_approached:
      MOV       DPH,#High(00000h)                ;
      MOV       DPL,#LOW(00000h)                 ;

isp_hw_write_loop:
      MOV       IFADRL,DPL                       ;
      MOV       IFADRH,DPH                       ;
      MOV       IFD,#055h                        ; ISP Program Data
      CALL      ixp_program                      ;

;     JNB       ixp_program_state,isp_hw_write_fail
                                                 ; Fail EQU 0

      JB        ixp_program_state,isp_hw_write_fail
                                                 ; Fail EQU 1
isp_hw_write_next:
      INC       DPTR                             ;

      MOV       A,DPH                            ;
      CJNE      A,#01Ch,isp_hw_write_loop        ;

      ANL       ISPCR,#~SWBS                     ;
      ORL       ISPCR,#SWRST                     ;

isp_hw_write_fail:
;           to do ...
      JMP       isp_hw_write_next                ;
```

C Code Example:

```
      unsigned short addr=0x0000;                //
      do{ IFADRH = (unsigned char)(addr >>8);
            IFADRL = (unsigned char)addr ;       //
            IFD = 0x55;                          //
            if(ixp_program() == Fail);           //
            {
//                to do ...
            }
      }while(++addr != 0x1C00);                  //
      ISPCR = SWRST;                             // Select AP Boot & SoftWave Reset
```

*(4). Required Function: ISP Sample code by Sofrware approached entrance*

Assembly Code Example:

```
      MOV       ISPCR,#(SWBS|SWRST)              ;
```

C Code Example:

```
      ISPCR = (SWBS | SWRST) ;                   //
```

*(5). Required Function: IAPLB Change to on-chip flash 4K boundary*

Assembly Code Example:

```
MOV       IFADRL,#IAPLB                    ;
MOV       IFD,#(HIGH(4096))&0xFE           ;
CALL      page_p_sfr_write                 ;
```

C Code Example:

```
IFADRL = IAPLB;                                  //
IFD = (((unsigned char)(4096 >> 8)) & 0xFE);     //
page_p_sfr_write();                              //
```

# 18. Page P SFR Access

**MG86FE/L104** builds a special SFR page (Page P) to store the control registers for MCU operation. These SFRs can be accessed by the ISP/IAP operation with different IFMT. In page P access, IFADRH must set to “00” and IFADRL indexes the SFR address in page P. If IFMT= 04H for Page P writing, the content in IFD will be loaded to the SFR in IFADRL indexed after the SCMD triggered. The SFRs in Page-P don’t support read function.

Following descriptions are the SFR function definition in Page P:

***IAPLB: IAP Low Boundary***

SFR Page = Page P Only
SFR Address = 0x03 RESET = 1111-1111

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| IAPLB | | | | | | | **0** |
| **W** | **W** | **W** | **W** | **W** | **W** | **W** | **W** |

Bit 7~0: The IAPLB determines the IAP-memory lower boundary.
To write IAPLB, MCU will put new IAPLB setting value in IFD firstly. And index IFADRL, select IMFT, enable ISPCR.ISPEN and then set SCMD. The IAPLB content has already finished the updated sequence.

The range of the IAP-memory is determined by IAPLB and the ISP Start address as listed below.
*IAP lower boundary = IAPLBx256, and*
*IAP higher boundary = ISP start address – 1.*

*For example, if IAPLB=0x08 and ISP start address is 0x0C00, then the IAP-memory range is located at 0x0800 ~ 0x0BFF.*

**Additional attention point, the IAP low boundary address must not be higher than ISP start address.**

***CKCON2: Clock Control Register 2***

SFR Page = Page P Only
SFR Address = 0x40 RESET = 0001-xx00

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| -- | -- | XTALE | IHRCOE | -- | -- | OSCS1 | OSCS0 |
| **W** | **W** | **W** | **W** | **W** | **W** | **W** | **W** |

Bit 7~6: Reserved. Software must write “0” on this bit when CKCON2 is written.

Bit 5: XTALE, external Crystal(XTAL) Enable. The default value is set by hardware option on clock source selection.
0: Disable XTAL oscillating circuit. In this case, XTAL2 and XTAL1 behave as Port 6.0 and Port 6.1.
1: Enable XTAL oscillating circuit. If this bit is set by CPU software, it needs **3 ms** to have stable output after XTALE ~~is~~ enabled.

Bit 4: IHRCOE, Internal High frequency RC Oscillator Enable. The default value is set by hardware option on clock source selection.
0: Disable internal high frequency RC oscillator.
1: Enable internal high frequency RC oscillator. If this bit is set by CPU software, it needs **32 us** to have stable output after IHRCOE enabled.

Bit 3~2: Reserved. Software must write “0” on these bits when CKCON2 is written.

Bit 1~0: OSC[1:0], OSCin source selection.

| OSCS[1:0] | OSCin source Selection |
|---|---|
| 0 0 | IHRCO |
| 0 1 | XTAL |
| 1 0 | ILRCO |
| 1 1 | ECKI, External Clock Input (P4.0) as OSCin. |

***PCON2: Power Control Register 2***

SFR Page = Page P Only

SFR Address = 0x44 RESET = x0xx-xx01

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| -- | AWBOD0 | -- | -- | -- | -- | BO0RE | **1** |
| **W** | **W** | **W** | **W** | **W** | **W** | **W** | **W** |

Bit 7: Reserved. Software must write “0” on this bit when PCON2 is written.

Bit 2: AWBOD0, Awaked BOD0 in PD mode.
0: BOD0 is disabled in power-down mode.
1: BOD0 keeps operation in power-down mode.

Bit 5~2: Reserved. Software must write “0” on these bits when PCON2 is written.

Bit 1: BO0RE, BOD0 Reset Enabled. Its initial value could be changed by hardware option, BO0REO.
0: Disable BOD0 to trigger a system reset when BOF0 is set.
1: Enable BOD0 to trigger a system reset when BOF0 is set by VDD meets 4.2V(E) or 2.4V(L).

Bit 0: Reserved for test. Software must write **“1”** on this bit when PCON2 is written.

***SPCON0: SFR Page Control 0***

SFR Page = Page P Only

SFR Address = 0x48 RESET = xxx0-x000

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| -- | -- | -- | WRCTL | -- | CKCTL0 | PWCTL1 | PWCTL0 |
| **W** | **W** | **W** | **W** | **W** | **W** | **W** | **W** |

Bit 7~5: Reserved. Software must write “0” on these bits when SPCON0 is written.

Bit 4: WRCTL. WDTCR SFR access Control.
If WRCTL is set, it will disable the WDTCR SFR modified in general Page. WDTCR in general Page only keeps the SFR read function. But software always owns the modification capability in SFR Page P.

Bit 3: Reserved. Software must write “0” on this bit when SPCON0 is written.

Bit 2: CKCTL0. CKCON0 SFR access Control.
If CKCTL0 is set, it will disable the CKCON0 SFR modified in general Page. CKCON0 in general Page only keeps the SFR read function. But software always owns the modification capability in SFR Page P.

Bit 1: PWCTL1. PCON1 SFR access Control.
If PWCTL1 is set, it will disable the PCON1 SFR modified in general Page. PCON1 in general Page only keeps the SFR read function. But software always owns the modification capability in SFR Page P.

Bit 0: PWCTL0. PCON0 SFR access Control.
If PWCTL0 is set, it will disable the PCON0 SFR modified in general Page. PCON0 in general Page only keeps the SFR read function. But software always owns the modification capability in SFR Page P.

***DCON0: Device Control 0***

SFR Page = Page P Only

SFR Address = 0x4C RESET = 00xx-xx1x

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| HSE | IAPO | -- | -- | -- | -- | RSTIO | -- |
| **W** | **W** | **W** | **W** | **W** | **W** | **W** | **W** |

Bit 7: HSE, High Speed operation Enable.
0: Disable high speed operation for MCU.
1: Enable high speed operation for MCU. ($F_{SYSCLK}$> 12MHz)

Bit 6: IAPO, IAP function Only.
0: Maintain IAP region to service IAP function and code execution.
1: Disable the code execution in IAP region and the region only service IAP function.

Bit 5~2: Reserved. Software must write “0” on these bits when DCON0 is written.

Bit 1: RSTIO, RST function on I/O,
0: Select I/O pad function for P36.
1: Select I/O pad function for RST.

Bit 0: Reserved. Software must write “0” on this bit when DCON0 is written.

## 18.1. Page-P Sample Code

*(1). Required Function: General function call of Page-P SFR Read*

Assembly Code Example:

```
_page_p_sfr_read:
page_p_sfr_read:
    MOV     IFADRH,000h
    MOV     IFMT,#(MS2|MS0)             ; PageP_Read=0x05

    ANL     ISPCR,#CFAIL                ;
    ORL     ISPCR,#ISPEN                ; Enable Function

    MOV     SCMD,#046h                  ;
    MOV     SCMD,#0B9h                  ;

    MOV     IFMT,#000h                  ; Flash_Standby=0x00
    ANL     ISPCR,#~ISPEN               ; Disable Function

    RET
```

C Code Example:

```
void page_p_sfr_read (void)
{
    IFADRH = 0x00;                      //

    ISPCR = ISPEN;                      // Enable Function
    IFMT = (MS0 | MS2);                 // PageP_Read=0x05

    SCMD = 0x46;                        //
    SCMD = 0xB9;                        //

    IFMT = Flash_Standby;               // Flash_Standby=0x00
    ISPCR &= ~ISPEN;
}
```

*(2). Required Function: General function call of Page-P SFR Write*

Assembly Code Example:

```
_page_p_sfr_write:
page_p_sfr_write:
    MOV     IFADRH,000h                 ;

    MOV     ISPCR,#ISPEN                ; Enable Function
    MOV     IFMT,#MS2                   ; PageP_Write=0x04

    MOV     SCMD,#046h                  ;
    MOV     SCMD,#0B9h                  ;

    MOV     IFMT,#000h                  ; Flash_Standby=0x00
    ANL     ISPCR,#~ISPEN               ; Disable Function

    RET
```

C Code Example:

```
void page_p_sfr_write (void)
{
    IFADRH = 0x00;

    ISPCR = ISPEN;                      // Enable Function
    IFMT = MS2;                         // PageP_Write=0x04

    SCMD = 0x46;                        //
    SCMD = 0xB9;                        //
```

```
    IFMT = Flash_Standby;                    // Flash_Standby=0x00
    ISPCR &= ~ISPEN;
}
```

*(3). Required Function: Enable PWCTL0 for PCON0.PD control in Page-P*

Assembly Code Example:

```
    MOV     IFADRL,#SPCON0              ;
    CALL    page_p_sfr_read             ;

    ORL     IFD,#PWCTL0                 ; Set PWCTL0
    CALL    page_p_sfr_write            ;

    MOV     IFD,PCON0                   ; Read PCON0

    ORL     IFD,#PD                     ; Write PCON0 and Power-Down
    MOV     IFADRL,#PCON0_P             ;
    CALL    page_p_sfr_write            ;
```

C Code Example:

```
    IFADRL = SPCON0;                    //
    page_p_sfr_read();                  //

    IFD |= PWCTL0;                      // Set PWCTL0
    page_p_sfr_write();                 //

    IFD = PCON0;                        // Read PCON0

    IFD |= PD;                          // Write PCON0
    IFADRL = PCON0_P;                   //
    page_p_sfr_write();                 //
```

*(4). Required Function: Enable CKCTL0 for SYSCLK divider (CKCON0) changed in Page-P*

Assembly Code Example:

```
    MOV     IFADRL,#SPCON0              ;
    CALL    page_p_sfr_read             ;

    ORL     IFD,#CKCTL0                 ; Set CKCTL0
    CALL    page_p_sfr_write            ;

    MOV     IFD,CKCON0                  ; Read CKCON0

    ORL     IFD,#(AFS | SCKS0)          ; Write CKCON0 and Set AFS
    MOV     IFADRL,#CKCON0_P            ; SYSCLK / 2
    CALL    page_p_sfr_write
```

C Code Example:

```
    IFADRL = SPCON0;                    //
    page_p_sfr_read ();                 //

    IFD |= CKCTL0;                      // Set CKCTL
    page_p_sfr_write();                 //

    IFD = CKCON0;                       // read CKCON0

    IFD |= (AFS | SCKS0);               //
    IFADRL = CKCON0_P;                  //
    page_p_sfr_write();                 // Write CKCON0
```

# 19. Auxiliary SFRs

***AUXR0: Auxiliary Register 0***

SFR Page = Normal
SFR Address = 0xA1 RESET = 0000-0000

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| P40OC1 | P40OC0 | P40FD | -- | **P1FS1** | **P1FS0** | INT1H | INT0H |
| **R/W** | **R/W** | **R/W** | **W** | **R/W** | **R/W** | **R/W** | **R/W** |

Bit 7~6: P4.0 function configured control bit 1 and 0. The two bits only act when internal RC oscillator (IHRCO or ILRCO) is selected for system clock source. In crystal mode, XTAL2 and XTAL1 are the alternated function for P4.0 and P4.1. In external clock input mode, P4.0 is the dedicated clock input pin. In internal oscillator condition, P4.0 provides the following selections for GPIO or clock source generator. When P40OC[1:0] index to non-P4.0 GPIO function, P4.0 will drive the on-chip RC oscillator (IHRCO or ILRCO) output to provide the clock source for other devices.

| P40OC[1:0] | P4.0 function | P4.0 I/O mode |
|---|---|---|
| 00 | P40 | By P4M0.0 |
| 01 | OSCin | By P4M0.0 |
| 10 | OSCin/2 | By P4M0.0 |
| 11 | OSCin/4 | By P4M0.0 |

For clock-out on P4.0 function, it is recommended to set P4M0.0 to “1” which selects P4.0 as push-push output mode.

Bit 5: P40FD, P4.0 Fast Driving.
0: P4.0 output with default driving.
1: P4.0 output with fast driving enabled. If P6.0 is configured to clock output, enable this bit when P6.0 output frequency is more than 12MHz at 5V application or more than 6MHz at 3V application.

Bit 4: Reserved. Software must write “0” on this bit when AUXR0 is written.

Bit 3~2: P1.4 and P1.5 alternated function selection.

| P1FS[1:0] | P1.4 | P1.5 |
|---|---|---|
| 00 | P1.4 | P1.5 |
| 01 | -- | Output for TXD1 |
| 10 | Input for nINT0 | Input for nINT1 |
| 11 | Input for T0 | Input for T1 |

Bit 1: INT1H, INT1 High/Rising trigger enable.
0: Remain INT1 triggered on low level or falling edge on P3.3.
1: Set INT1 triggered on high level or rising edge on P3.3.

Bit 0: INT0H, INT0 High/Rising trigger enable.
0: Remain INT0 triggered on low level or falling edge on P3.2.
1: Set INT0 triggered on high level or rising edge on P3.2.

***AUXR1: Auxiliary Control Register 1***

SFR Page = Normal
SFR Address = 0xA2 RESET = 0000-0000

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| RTX3E | RTX2E | RTX1E | RTX0E | -- | -- | RXCS1 | RXCS0 |
| **R/W** | **R/W** | **R/W** | **R/W** | **W** | **W** | **R/W** | **R/W** |

Bit 7: RTX3E, RXD Repeat to TXD3 enabled.
0: Disable RTX3E function on TXD3 (P1.5).
1: Enable the RXD repeat mechanism on TXD3 (P1.5).

Bit 6: RTX2E, RXD Repeat to TXD2 enabled.
0: Disable RTX2E function on TXD2 (P1.3).
1: Enable the RXD repeat mechanism on TXD3 (P1.3).

Bit 5: RTX1E, RXD Repeat to TXD1 enabled.
0: Disable RTX1E function on TXD1 (P1.2).
1: Enable the RXD repeat mechanism on TXD1 (P1.2).

Bit 4: RTX0E, RXD Repeat to TXD0 enabled.
0: Disable RTX0E function on TXD0 (P3.1).
1: Enable the RXD repeat mechanism on TXD0 (P3.1).

Bit 3~2: Reserved. Software must write “0” on these bits when AUXR1 is written.

Bit 1~0: RXD channel selection.

| RXCS[1:0] | RXD channel selection |
|---|---|
| 00 | P3.0 |
| 01 | P3.2 |
| 10 | P3.3 |
| 11 | P1.4 |

***AUXR2: Auxiliary Register 2***

SFR Page = Normal
SFR Address = 0xA3 RESET = 0000-0000

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| URXR | BTI | URM0X6 | SMOD2 | T1X12 | T0X12 | T1CKOE | T0CKOE |
| **R/W** | **R/W** | **R/W** | **R/W** | **R/W** | **R/W** | **R/W** | **R/W** |

Bit 7: URXR, Serial Port test mode. Software must write “0” on this bit when AUXR2 is written.

Bit 6: BTI, Block TI in Serial Port Interrupt.
0: Retain the TI to be a source of Serial Port Interrupt.
1: Block TI to be a source of Serial Port Interrupt.

Bit 5: URM0X6, Serial Port mode 0 baud rate selector.
0: Clear to select SYSCLK/12 as the baud rate for UART Mode 0.
1: Set to select SYSCLK/2 as the baud rate for UART Mode 0.

Bit 4: SMOD2, extra X2/X4 baud rate selector.
0: Disable extra X2/X4 baud rate.
1: Enable extra X2/X4 baud rate.

Bit 3: T1X12, Timer 1 clock source selector while C/T=0.
0: Clear to select SYSCLK/12.
1: Set to select SYSCLK as the clock source.

Bit 2: T0X12, Timer 0 clock source selector while C/T=0.
0: Clear to select SYSCLK/12.
1: Set to select SYSCLK as the clock source.

Bit 1: T1CKOE, Timer 1 Clock Output Enable.
0: Disable Timer 1 clock output.
1: Enable Timer 1 clock output on P3.5.

Bit 0: T0CKOE, Timer 0 Clock Output Enable.
0: Disable Timer 0 clock output.
1: Enable Timer 0 clock output on P3.4.

# 20. Hardware Option

The MCU's Hardware Option defines the device behavior which cannot be programmed or controlled by software. The hardware options can only be programmed by a Universal Programmer, the "Megawin 8051 Writer U1" or the "Megawin 8051 ICE Adapter" (The ICE adapter also supports ICP programming function. Refer Section "21.4 ICP Interface Circuit"). After whole-chip erased, all the hardware options are left in "disabled" state and there is no ISP-memory and IAP-memory configured. The **MG86FE/L104** has the following Hardware Options:

**LOCK:**
☑: Enabled. Code dumped on a universal Writer or Programmer is locked to 0xFF for security.
☐: Disabled. Not locked.

**ISP-memory Space**:
The ISP-memory space is specified by its starting address. And, its higher boundary is limited by the Flash end address, i.e., 0x0FFF. The following table lists the ISP space option in this chip. In default setting, **MG86FE/L104** ISP space is configured to 1K that had been embedded Megawin proprietary ISP code to perform In-System-Programming through Megawin 1-Line ISP protocol.

| ISP-memory Size | ISP Start Address |
|---|---|
| 3.5K bytes | 0x0200 |
| 3K bytes | 0x0400 |
| 2.5K bytes | 0x0600 |
| 2K bytes | 0x0800 |
| 1.5K bytes | 0x0A00 |
| 1K bytes | 0x0C00 |
| 0.5K bytes | 0x0E00 |
| No ISP Space | -- |

**HWBS:**
☑: Enabled. When powered up, MCU will boot from ISP-memory if ISP-memory is configured.
☐: Disabled. MCU always boots from AP-memory.

**HWBS2:**
☑: Enabled. Not only power-up but also any reset will cause MCU to boot from ISP-memory if ISP-memory is configured.
☐: Disabled. Where MCU boots from is determined by HWBS.

**IAP-memory Space**:
The IAP-memory space specifies the user defined IAP space. The IAP-memory Space can be configured by hardware option or MCU software by modifying IAPLB. In default, it is configured to 1K bytes.

**BO0REO:**
☑: Enabled. BOD0 will trigger a RESET event to CPU on AP program start address. (2.2V)
☐: Disabled. BOD0 can not trigger a RESET to CPU.

**WRENO:**
☑: Enabled. Set WDTCR.WREN to enable a system reset function by WDTF.
☐: Disabled. Clear WDTCR.WREN to disable the system reset function by WDTF.

**NSWDT:** Non-Stopped WDT
☑: Enabled. Set WDTCR.NSW to enable the WDT running in power down mode (watch mode).
☐: Disabled. Clear WDTCR.NSW to disable the WDT running in power down mode (disable Watch mode).

**HWENW:** Hardware loaded for "ENW" of WDTCR.
☑: Enabled. Enable WDT and load the content of WRENO, NSWDT, HWWIDL and HWPS2~0 to WDTCR after power-on.

☐: Disabled. WDT is not enabled automatically after power-on.

**HWWIDL, HWPS2, HWPS1, HWPS0:**
When HWENW is enabled, the content on these four fused bits will be loaded to WDTCR SFR after power-on.

**WDSFWP:**
☑: Enabled. The WDT SFRs, WREN, NSW, WIDL, PS2, PS1 and PS0 in WDTCR, will be write-protected.
☐: Disabled. The WDT SFRs, WREN, NSW, WIDL, PS2, PS1 and PS0 in WDTCR, are free for writing of software.

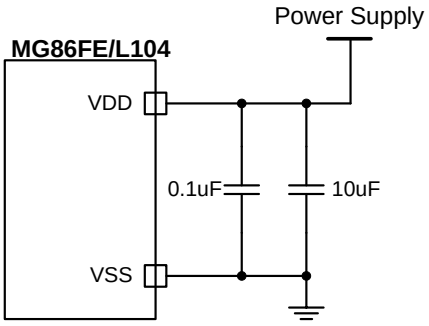
**P36EN:**
☑: Enabled. RSTIO (DCON0.1) will be cleared and P3.6 function behaves on RST pin.
☐: Disabled. RSTIO (DCON0.1) will be set and reserve RST pin funciton.

# 21. Application Notes

## 21.1. Power Supply Circuit

To have the **MG86FE/L104** work with power supply varying from 4.2V to 5.5V for E-type and from 2.4V to 3.6V for L-type, adding some external decoupling and bypass capacitors is necessary, as shown in Figure 21–1.

Figure 21–1. Power Supplied Circuit



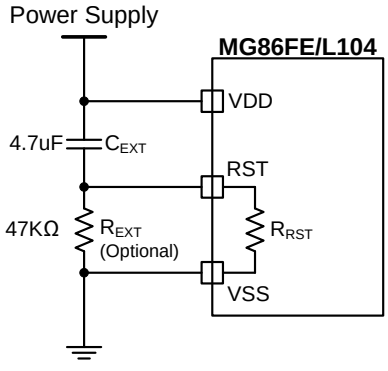

## 21.2. Reset Circuit

Normally, the power-on reset can be successfully generated during power-up. However, to further ensure the MCU a reliable reset during power-up, the external reset is necessary. Figure 21–2 shows the external reset circuit, which consists of a capacitor $C_{EXT}$ connected to VDD (power supply) and a resistor $R_{EXT}$ connected to VSS (ground).

In general, $R_{EXT}$ is optional because the RST pin has an internal pull-down resistor ($R_{RST}$). This internal diffused resistor to VSS permits a power-up reset using only an external capacitor $C_{EXT}$ to VDD.

See Section "22.2 DC Characteristics" for $R_{RST}$ value.

Figure 21–2. Reset Circuit



## 21.3. XTAL Oscillating Circuit

To achieve successful and exact oscillating (up to 24MHz), the capacitors C1 and C2 are necessary, as shown in Figure 21–3. Normally, C1 and C2 have the same value. Table 21–1 lists the C1 & C2 value for the different frequency crystal application.

Figure 21–3. XTAL Oscillating Circuit

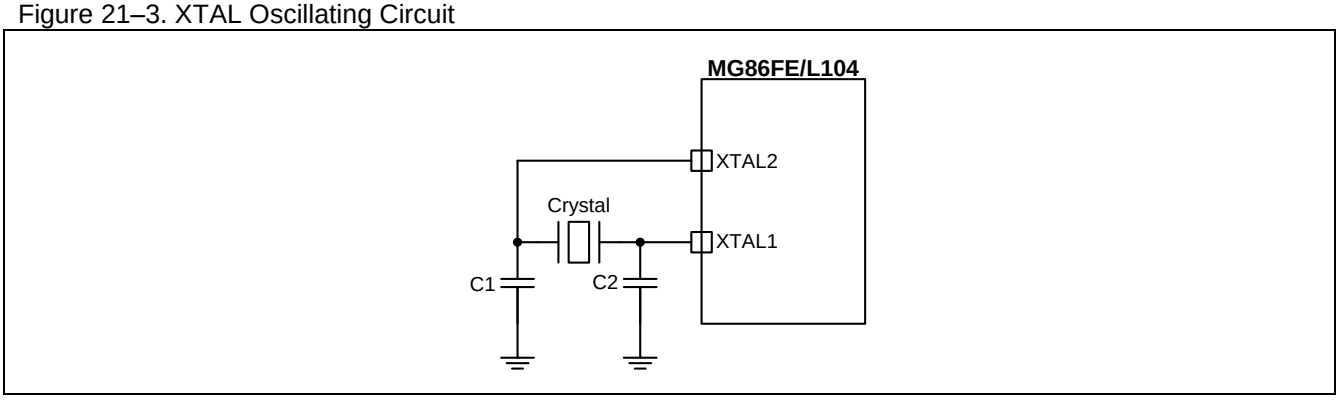



Table 21–1. Reference Capacitance of C1 & C2 for crystal oscillating circuit

| Crystal | C1, C2 Capacitance |
|---|---|
| 16MHz ~ 25MHz | 10pF |
| 6MHz ~ 16MHz | 15pF |
| 2MHz ~ 6MHz | 33pF |

## 21.4. ICP Interface Circuit

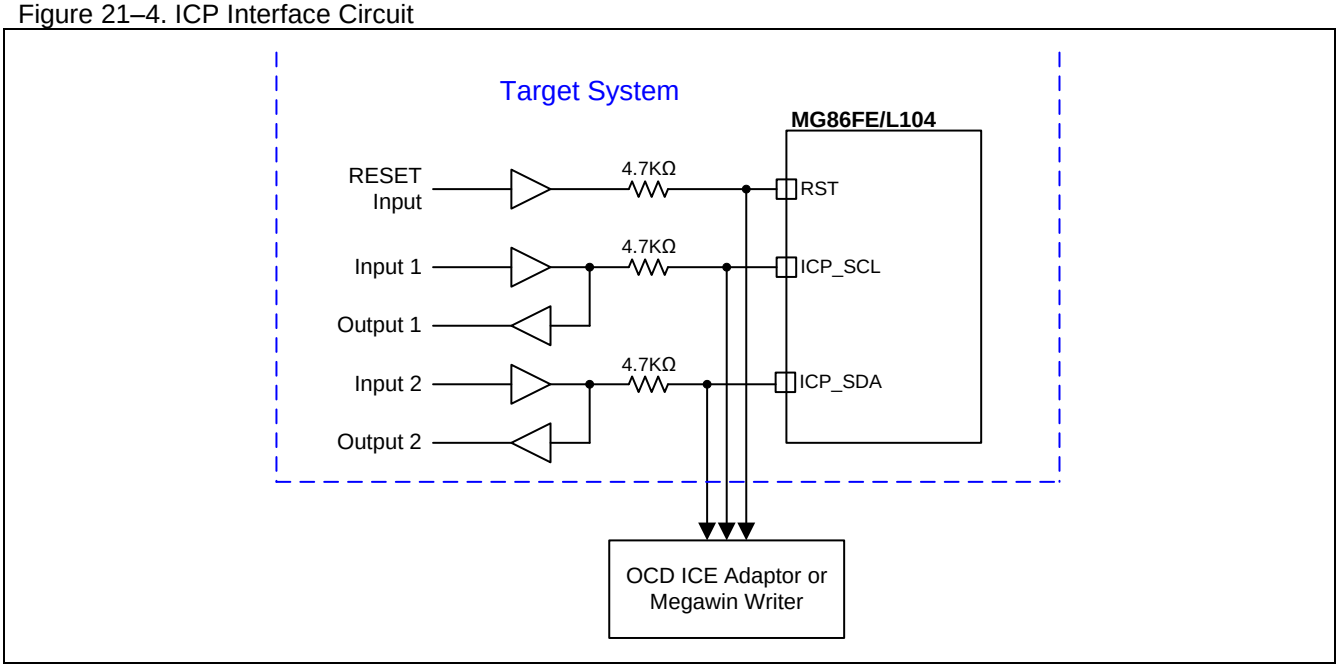
**MG86FE/L104** devices include an on-chip Megawin proprietary programming interface to allow In-Chip-Programming (ICP) with the production part installed in the end application. The ICP interface uses a clock signal (ICP_SCL) and a bi-directional data signal (ICP_SDA) to perform the device programming from a host instruction.

The ICP interface allows the ICP_SCL/ICP_SDA pins to be shared with user functions so that In-Chip Flash Programming function could be performed. This is practicable because ICP communication is performed when the device is in the halt state, where the on-chip peripherals and user software are stalled. In this halted state, the ICP interface can safely 'borrow' the ICP_SCL (P1.6) and ICP_SDA (P1.5) pins. In most applications, external resistors are required to isolate ICP interface traffic from the user application. A typical isolation configuration is shown in Figure 21–4.

***It is strongly recommended to build the ICP interface circuit on target system. It will reserve the whole capability for software programming and device options configured.***

*Note: MG86FE/L104AS8 does not support ICP interface.*

Figure 21–4. ICP Interface Circuit



# 22. Electrical Characteristics

## 22.1. Absolute Maximum Rating

**For MG86FE104:**

| Parameter | Rating | Unit |
|---|---|---|
| Ambient temperature under bias | -40 ~ +85 | °C |
| Storage temperature | -65 ~ + 150 | °C |
| Voltage on any Port I/O Pin or RST with respect to VSS | -0.5 ~ VDD + 0.5 | V |
| Voltage on VDD with respect to VSS | -0.5 ~ +6.0 | V |
| Maximum total current through VDD and VSS | 200 | mA |
| Maximum output current sunk by any Port pin | 40 | mA |

*Note: stresses above those listed under “Absolute Maximum Ratings” may cause permanent damage to the device. This is a stress rating only and functional operation of the devices at those or any other conditions above those indicated in the operation listings of this specification is not implied. Exposure to maximum rating conditions for extended periods may affect device reliability.

**For MG86FL104:**

| Parameter | Rating | Unit |
|---|---|---|
| Ambient temperature under bias | -40 ~ +85 | °C |
| Storage temperature | -65 ~ + 150 | °C |
| Voltage on any Port I/O Pin or RESET with respect to Ground | -0.3 ~ VDD + 0.3 | V |
| Voltage on VDD with respect to Ground | -0.3 ~ +4.2 | V |
| Maximum total current through VDD and Ground | 200 | mA |
| Maximum output current sunk by any Port pin | 40 | mA |

*Note: stresses above those listed under “Absolute Maximum Ratings” may cause permanent damage to the device. This is a stress rating only and functional operation of the devices at those or any other conditions above those indicated in the operation listings of this specification is not implied. Exposure to maximum rating conditions for extended periods may affect device reliability.

## 22.2. DC Characteristics

**For MG86FE104:**

VDD = 5.0V±10%, VSS = 0V, $T_A$ = 25 ℃ and execute NOP for each machine cycle, unless otherwise specified

| Symbol | Parameter | Test Condition | Limits | | | Unit |
|---|---|---|---|---|---|---|
| | | | min | typ | max | |
| **Input/Output Characteristics** | | | | | | |
| $V_{IH1}$ | Input High voltage (All I/O Ports) | Except P4.0, P4.1 | 2.0 | | | V |
| $V_{IH2}$ | Input High voltage (RST, P4.0, P4.1) | | 3.5 | | | V |
| $V_{IL1}$ | Input Low voltage (All I/O Ports) | Except P6.0, P6.1 | | | 0.8 | V |
| $V_{IL2}$ | Input Low voltage (RST, P6.0, P6.1) | | | | 1.0 | V |
| $I_{IH}$ | Input High Leakage current (All I/O Ports) | $V_{PIN}$ = VDD | | 0 | 10 | uA |
| $I_{IL1}$ | Logic 0 input current (P3 in quasi-mode) | $V_{PIN}$ = 0.4V | | 28 | 60 | uA |
| $I_{IL2}$ | Logic 0 input current (All Input only or open-drain Ports) | $V_{PIN}$ = 0.4V | | 0 | 10 | uA |
| $I_{H2L}$ | Logic 1 to 0 input transition current (P3 in quasi-mode) | $V_{PIN}$ =1.8V | | 330 | 500 | uA |
| $I_{OH1}$ | Output High current (P3 in quasi-Mode) | $V_{PIN}$ =2.4V | 150 | 200 | | uA |
| $I_{OH2}$ | Output High current (All push-pull output ports) | $V_{PIN}$ =2.4V | 12 | | | mA |
| $I_{OL1}$ | Output Low current (All I/O Ports) | $V_{PIN}$ =0.4V | 12 | | | mA |
| $R_{RST}$ | Internal reset pull-down resistance | | | 77 | | Kohm |
| **Power Consumption** | | | | | | |
| $I_{OP1}$ | Normal mode operating current | SYSCLK = 24MHz @ IHRCO | | 8.5 | | mA |
| $I_{OP2}$ | | SYSCLK = 12MHz @ IHRCO | | 5.9 | | mA |
| $I_{OP3}$ | | SYSCLK = 6MHz @ IHRCO & HSE = 0 | | 4.1 | | mA |
| $I_{OP4}$ | | SYSCLK = 3MHz @ IHRCO & HSE = 0 | | 2.4 | | mA |
| $I_{OP5}$ | | SYSCLK = 24MHz @ XTAL | | 10 | | mA |
| $I_{OP6}$ | | SYSCLK = 12MHz @ XTAL | | 6.9 | | mA |
| $I_{OP7}$ | | SYSCLK = 6MHz @ XTAL & HSE = 0 | | 5 | | mA |
| $I_{OP8}$ | | SYSCLK = 2MHz @ XTAL & HSE = 0 | | 2.5 | | mA |
| $I_{OPS1}$ | Slow mode operating current | SYSCLK = 24MHz/128 @ IHRCO | | 0.86 | | mA |
| $I_{IDLE1}$ | Idle mode operating current | SYSCLK = 24MHz @ IHRCO | | 2.1 | | mA |
| $I_{IDLE2}$ | | SYSCLK = 24MHz @ XTAL | | 3.6 | | mA |
| $I_{IDLE3}$ | | SYSCLK = 24MHz/128 @ IHRCO | | 0.77 | | mA |
| $I_{IDLE4}$ | | SYSCLK = 24MHz/128 @ XTAL | | 2.2 | | mA |
| $I_{IDLE5}$ | | SYSCLK = 64KHz @ ILRCO | | 16 | | uA |
| $I_{SUB1}$ | Sub-clock mode operating current | SYSCLK = 64KHz @ ILRCO, BOD0 disabled & HSE = 0 | | 45 | | uA |

| | | | | | | |
|---|---|---|---|---|---|---|
| $I_{SUB2}$ | | SYSCLK = 64KHz/128 @ ILRCO, BOD0 disabled & HSE = 0 | | 13 | | uA |
| $I_{WAT}$ | Watch mode operating current | WDT = 64KHz @ ILRCO in PD mode | | 2.5 | | uA |
| $I_{MON1}$ | Monitor Mode operating current | BOD0 enabled in PD mode | | 10 | | uA |
| **BOD0 Characteristics** | | | | | | |
| $V_{BOD0}$ | BOD0 detection level | $T_A$ = -40℃ to +85℃ | 3.8(1) | 4.0 | 4.2(1) | V |
| **Operating Condition** | | | | | | |
| $V_{PSR}$ | Power-on Slop Rate | $T_A$ = -40℃ to +85℃ | 0.05 | | | V/ms |
| $V_{OP1}$ | Operating Speed 0–25MHz | $T_A$ = -40℃ to +85℃ | 4.5 | | 5.5 | V |
| $V_{OP2}$ | Operating Speed 0-12MHz | $T_A$ = -40℃ to +85℃ | 4.2 | | 5.5 | V |

(1) Data based on characterization results, not tested in production.

**For MG86FL104:**

VDD = 3.3V±10%, VSS = 0V, $T_A$ = 25 ℃ and execute NOP for each machine cycle, unless otherwise specified

| Symbol | Parameter | Test Condition | Limits | | | Unit |
|---|---|---|---|---|---|---|
| | | | min | typ | max | |
| **Input/Output Characteristics** | | | | | | |
| $V_{IH1}$ | Input High voltage (All I/O Ports) | Except P4.0, P4.1 | 2.0 | | | V |
| $V_{IH2}$ | Input High voltage (RST, P4.0, P4.1) | | 2.4 | | | V |
| $V_{IL1}$ | Input Low voltage (All I/O Ports) | Except P6.0, P6.1 | | | 0.8 | V |
| $V_{IL2}$ | Input Low voltage (RST, P6.0, P6.1) | | | | 0.8 | V |
| $I_{IH}$ | Input High Leakage current (All I/O Ports) | $V_{PIN}$ = VDD | | 0 | 10 | uA |
| $I_{IL1}$ | Logic 0 input current (P3 in quasi-mode) | $V_{PIN}$ = 0.4V | | 10 | 30 | uA |
| $I_{IL2}$ | Logic 0 input current (All Input only or open-drain Ports) | $V_{PIN}$ = 0.4V | | 0 | 10 | uA |
| $I_{H2L}$ | Logic 1 to 0 input transition current (P3 in quasi-mode) | $V_{PIN}$ =1.8V | | 120 | 250 | uA |
| $I_{OH1}$ | Output High current (P3 in quasi-Mode) | $V_{PIN}$ =2.4V | 40 | 80 | | uA |
| $I_{OH2}$ | Output High current (All push-pull output ports) | $V_{PIN}$ =2.4V | 8 | | | mA |
| $I_{OL1}$ | Output Low current (All I/O Ports) | $V_{PIN}$ =0.4V | 8 | | | mA |
| $R_{RST}$ | Internal reset pull-down resistance | | | 93 | | Kohm |
| **Power Consumption** | | | | | | |
| $I_{OP1}$ | Normal mode operating current | SYSCLK = 24MHz @ IHRCO | | 8.8 | | mA |
| $I_{OP2}$ | | SYSCLK = 12MHz @ IHRCO | | 5.7 | | mA |
| $I_{OP3}$ | | SYSCLK = 6MHz @ IHRCO & HSE = 0 | | 4 | | mA |
| $I_{OP4}$ | | SYSCLK = 3MHz @ IHRCO & HSE = 0 | | 2.4 | | mA |
| $I_{OP5}$ | | SYSCLK = 24MHz @ XTAL | | 8.9 | | mA |
| $I_{OP6}$ | | SYSCLK = 12MHz @ XTAL | | 5.6 | | mA |
| $I_{OP7}$ | | SYSCLK = 6MHz @ XTAL & HSE = 0 | | 3.8 | | mA |
| $I_{OP8}$ | | SYSCLK = 2MHz @ XTAL & HSE = 0 | | 1.6 | | mA |
| $I_{OPS1}$ | Slow mode operating current | SYSCLK = 24MHz/128 @ IHRCO | | 0.86 | | mA |
| $I_{IDLE1}$ | Idle mode operating current | SYSCLK = 24MHz @ IHRCO | | 2.1 | | mA |
| $I_{IDLE2}$ | | SYSCLK = 24MHz @ XTAL | | 2.3 | | mA |
| $I_{IDLE3}$ | | SYSCLK = 24MHz/128 @ IHRCO | | 0.77 | | mA |
| $I_{IDLE4}$ | | SYSCLK = 24MHz/128 @ XTAL | | 0.88 | | mA |
| $I_{IDLE5}$ | | SYSCLK = 64KHz @ ILRCO | | 16 | | uA |
| $I_{SUB1}$ | Sub-clock mode operating current | SYSCLK = 64KHz @ ILRCO, BOD0 disabled & HSE = 0 | | 45 | | uA |

| | | | | | | |
|---|---|---|---|---|---|---|
| $I_{SUB2}$ | | SYSCLK = 64KHz/128 @ ILRCO, BOD0 disabled & HSE = 0 | | 13 | | uA |
| $I_{WAT}$ | Watch mode operating current | WDT = 64KHz @ ILRCO in PD mode | | 2.5 | | uA |
| $I_{MON1}$ | Monitor Mode operating current | BOD0 enabled in PD mode | | 10 | | uA |
| **BOD0 Characteristics** | | | | | | |
| $V_{BOD0}$ | BOD0 detection level | $T_A$ = -40℃ to +85℃ | 2.45(1) | 2.6 | 2.75(1) | V |
| **Operating Condition** | | | | | | |
| $V_{PSR}$ | Power-on Slop Rate | $T_A$ = -40℃ to +85℃ | 0.05 | | | V/ms |
| $V_{OP1}$ | Operating Speed 0–25MHz | $T_A$ = -40℃ to +85℃ | 2.7 | | 3.6 | V |
| $V_{OP2}$ | Operating Speed 0-12MHz | $T_A$ = -40℃ to +85℃ | 2.4 | | 3.6 | V |

(1) Data based on characterization results, not tested in production.

## 22.3. External Clock Characteristics

**For MG86FE104:**

VDD = 4.5V ~ 5.5V, VSS = 0V, $T_A$ = -40℃ to +85℃, unless otherwise specified

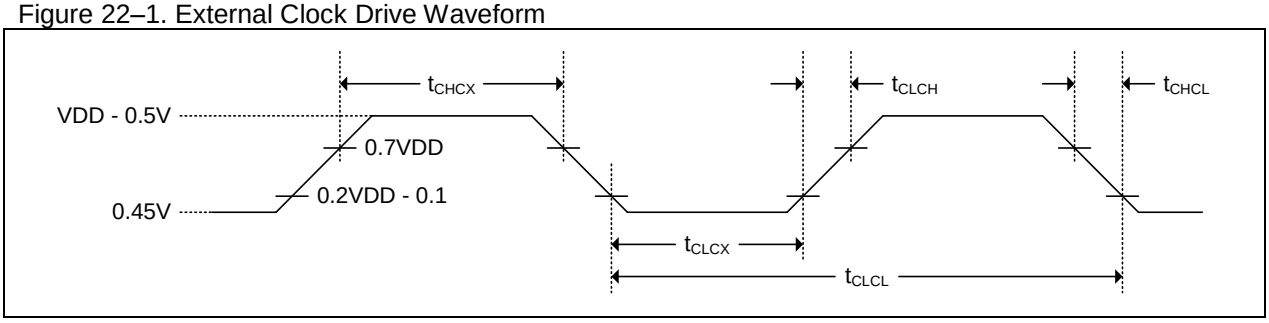
| Symbol | Parameter | Oscillator | | | | Unit |
|---|---|---|---|---|---|---|
| | | Crystal Mode | | ECKI Mode | | |
| | | Min. | Max | Min. | Max | |
| $1/t_{CLCL}$ | Oscillator Frequency | 2 | 25 | 0 | 25 | MHz |
| $1/t_{CLCL}$ | Oscillator Frequency (VDD = 4.2V ~ 5.5V) | 2 | 12 | 0 | 12 | MHz |
| $t_{CLCL}$ | Clock Period | 40 | | 40 | | ns |
| $t_{CHCX}$ | High Time | 0.4T | 0.6T | 0.4T | 0.6T | $t_{CLCL}$ |
| $t_{CLCX}$ | Low Time | 0.4T | 0.6T | 0.4T | 0.6T | $t_{CLCL}$ |
| $t_{CLCH}$ | Rise Time | | 5 | | 5 | ns |
| $t_{CHCL}$ | Fall Time | | 5 | | 5 | ns |

**For MG86FL104:**

VDD = 2.7V ~ 3.6V, VSS = 0V, $T_A$ = -40℃ to +85℃, unless otherwise specified

| Symbol | Parameter | Oscillator | | | | Unit |
|---|---|---|---|---|---|---|
| | | Crystal Mode | | ECKI Mode | | |
| | | Min. | Max | Min. | Max | |
| $1/t_{CLCL}$ | Oscillator Frequency | 2 | 25 | 0 | 25 | MHz |
| $1/t_{CLCL}$ | Oscillator Frequency (VDD = 2.4V ~ 3.6V) | 2 | 12 | 0 | 12 | MHz |
| $t_{CLCL}$ | Clock Period | 40 | | 40 | | ns |
| $t_{CHCX}$ | High Time | 0.4T | 0.6T | 0.4T | 0.6T | $t_{CLCL}$ |
| $t_{CLCX}$ | Low Time | 0.4T | 0.6T | 0.4T | 0.6T | $t_{CLCL}$ |
| $t_{CLCH}$ | Rise Time | | 5 | | 5 | ns |
| $t_{CHCL}$ | Fall Time | | 5 | | 5 | ns |

Figure 22–1. External Clock Drive Waveform



## 22.4. IHRCO Characteristics

**For MG86FE104:**

| Parameter | Test Condition | Limits | | | Unit |
|---|---|---|---|---|---|
| | | min | typ | max | |
| Supply Voltage | | 4.5 | | 5.5 | V |
| IHRCO Frequency | TA = +25℃, AFS = 0 | | 24 | | MHz |
| | TA = +25℃, AFS = 1 | | 22.118 | | MHz |
| IHRCO Frequency Deviation (factory calibrated) | TA = +25℃ | -1.0 | | +1.0 | % |
| | TA = -40℃ to +85℃ | -4.0(1) | | +4.0(1) | % |
| IHRCO Start-up Time | TA = -40℃ to +85℃ | | | 32(1) | us |
| IHRCO Power Consumption | TA = +25℃, VDD=5.0V | | 770 | | uA |

(1) Data based on characterization results, not tested in production.

**For MG86FL104:**

| Parameter | Test Condition | Limits | | | Unit |
|---|---|---|---|---|---|
| | | min | typ | max | |
| Supply Voltage | | 2.7 | | 3.6 | V |
| IHRCO Frequency | TA = +25℃, AFS = 0 | | 24 | | MHz |
| | TA = +25℃, AFS = 1 | | 22.118 | | MHz |
| IHRCO Frequency Deviation (factory calibrated) | TA = +25℃ | -1.0 | | +1.0 | % |
| | TA = -40℃ to +85℃ | -4.0(1) | | +4.0(1) | % |
| IHRCO Start-up Time | TA = -40℃ to +85℃ | | | 32(1) | us |
| IHRCO Power Consumption | TA = +25℃, VDD=5.0V | | 770 | | uA |

(1) Data based on characterization results, not tested in production.

## 22.5. ILRCO Characteristics

**For MG86FE104:**

| Parameter | Test Condition | Limits | | | Unit |
|---|---|---|---|---|---|
| | | min | typ | max | |
| Supply Voltage | | 4.2 | | 5.5 | V |
| ILRCO Frequency | TA = +25℃ | | 64 | | KHz |
| ILRCO Frequency Deviation | TA = +25℃ | -20(1) | | +20(1) | % |
| | TA = -40℃ to +85℃ | -40(1) | | +40(1) | % |

(1) Data based on characterization results, not tested in production.

**For MG86FL104:**

| Parameter | Test Condition | Limits | | | Unit |
|---|---|---|---|---|---|
| | | min | typ | max | |
| Supply Voltage | | 2.4 | | 3.6 | V |
| ILRCO Frequency | TA = +25℃ | | 64 | | KHz |
| ILRCO Frequency Deviation | TA = +25℃ | -20(1) | | +20(1) | % |
| | TA = -40℃ to +85℃ | -40(1) | | +40(1) | % |

(1) Data based on characterization results, not tested in production.

## 22.6. Flash Characteristics

**For MG86FE104:**

| Parameter | Test Condition | Limits | | | Unit |
|---|---|---|---|---|---|
| | | min | typ | max | |
| Supply Voltage | TA = -40℃ to +85℃ | 4.2 | | 5.5 | V |
| Flash Write (Erase/Program) Voltage | TA = -40℃ to +85℃ | 4.5 | | 5.5 | V |
| Flash Erase/Program Cycle | TA = -40℃ to +85℃ | 100 | | | times |
| Flash Data Retention | TA = +25℃ | 100 | | | year |

**For MG86FL104:**

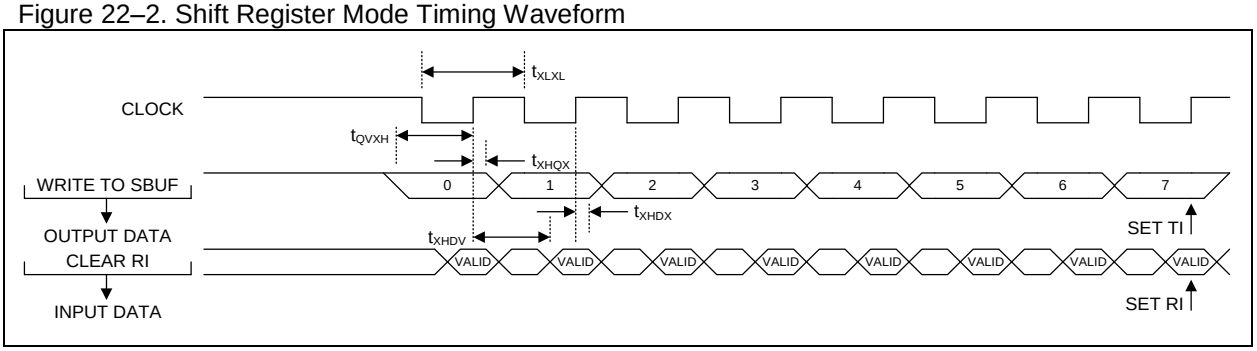
| Parameter | Test Condition | Limits | | | Unit |
|---|---|---|---|---|---|
| | | min | typ | max | |
| Supply Voltage | TA = -40℃ to +85℃ | 2.4 | | 3.6 | V |
| Flash Write (Erase/Program) Voltage | TA = -40℃ to +85℃ | 2.7 | | 3.6 | V |
| Flash Erase/Program Cycle | TA = -40℃ to +85℃ | 100 | | | times |
| Flash Data Retention | TA = +25℃ | 100 | | | year |

## 22.7. Serial Port Timing Characteristics

**For MG86FE104:** VDD = 5.0V±10%, VSS = 0V, $T_A$ = -40℃ to +85℃, unless otherwise specified
**For MG86FL104:** VDD = 3.3V±10%, VSS = 0V, $T_A$ = -40℃ to +85℃, unless otherwise specified

| Symbol | Parameter | URM0X6 = 0 | | URM0X6 = 1 | | Unit |
|---|---|---|---|---|---|---|
| | | Min. | Max | Min. | Max | |
| $t_{XLXL}$ | Serial Port Clock Cycle Time | 12T | | 2T | | $T_{SYSCLK}$ |
| $t_{QVXH}$ | Output Data Setup to Clock Rising Edge | 10T-20 | | T-20 | | ns |
| $t_{XHQX}$ | Output Data Hold after Clock Rising Edge | T-10 | | T-10 | | ns |
| $t_{XHDX}$ | Input Data Hold after Clock Rising Edge | 0 | | 0 | | ns |
| $t_{XHDV}$ | Clock Rising Edge to Input Data Valid | | 10T-20 | | 2T-20 | ns |

Figure 22–2. Shift Register Mode Timing Waveform



# 23. Instruction Set

Table 23–1. Instruction Set

| MNEMONIC | DESCRIPTION | BYTE | EXECUTION Cycles |
|---|---|---|---|
| ***DATA TRASFER*** | | | |
| MOV A,Rn | Move register to Acc | 1 | 1 |
| MOV A,direct | Move direct byte o Acc | 2 | 2 |
| MOV A,@Ri | Move indirect RAM to Acc | 1 | 2 |
| MOV A,#data | Move immediate data to Acc | 2 | 2 |
| MOV Rn,A | Move Acc to register | 1 | 2 |
| MOV Rn,direct | Move direct byte to register | 2 | 4 |
| MOV Rn,#data | Move immediate data to register | 2 | 2 |
| MOV direct,A | Move Acc to direct byte | 2 | 3 |
| MOV direct,Rn | Move register to direct byte | 2 | 3 |
| MOV direct,direct | Move direct byte to direct byte | 3 | 4 |
| MOV direct,@Ri | Move indirect RAM to direct byte | 2 | 4 |
| MOV direct,#data | Move immediate data to direct byte | 3 | 3 |
| MOV @Ri,A | Move Acc to indirect RAM | 1 | 3 |
| MOV @Ri,direct | Move direct byte to indirect RAM | 2 | 3 |
| MOV @Ri,#data | Move immediate data to indirect RAM | 2 | 3 |
| MOV DPTR,#data16 | Load DPTR with a 16-bit constant | 3 | 3 |
| MOVC A,@A+DPTR | Move code byte relative to DPTR to Acc | 1 | 4 |
| MOVC A,@A+PC | Move code byte relative to PC to Acc | 1 | 4 |
| MOVX A,@Ri | Move on-chip auxiliary RAM(8-bit address) to Acc | 1 | Not Support |
| MOVX A,@DPTR | Move on-chip auxiliary RAM(16-bit address) to Acc | 1 | Not Support |
| MOVX @Ri,A | Move Acc to on-chip auxiliary RAM(8-bit address) | 1 | Not Support |
| MOVX @DPTR,A | Move Acc to on-chip auxiliary RAM(16-bit address) | 1 | Not Support |
| MOVX A,@Ri | Move external RAM(8-bit address) to Acc | 1 | Not Support |
| MOVX A,@DPTR | Move external RAM(16-bit address) to Acc | 1 | Not Support |
| MOVX @Ri,A | Move Acc to external RAM(8-bit address) | 1 | Not Support |
| MOVX @DPTR,A | Move Acc to external RAM(16-bit address) | 1 | Not Support |
| PUSH direct | Push direct byte onto Stack | 2 | 4 |
| POP direct | Pop direct byte from Stack | 2 | 3 |
| XCH A,Rn | Exchange register with Acc | 1 | 3 |
| XCH A,direct | Exchange direct byte with Acc | 2 | 4 |
| XCH A,@Ri | Exchange indirect RAM with Acc | 1 | 4 |
| XCHD A,@Ri | Exchange low-order digit indirect RAM with Acc | 1 | 4 |
| ***ARITHEMATIC OPERATIONS*** | | | |
| ADD A,Rn | Add register to Acc | 1 | 2 |
| ADD A,direct | Add direct byte to Acc | 2 | 3 |
| ADD A,@Ri | Add indirect RAM to Acc | 1 | 3 |
| ADD A,#data | Add immediate data to Acc | 2 | 2 |
| ADDC A,Rn | Add register to Acc with Carry | 1 | 2 |
| ADDC A,direct | Add direct byte to Acc with Carry | 2 | 3 |
| ADDC A,@Ri | Add indirect RAM to Acc with Carry | 1 | 3 |
| ADDC A,#data | Add immediate data to Acc with Carry | 2 | 2 |
| SUBB A,Rn | Subtract register from Acc with borrow | 1 | 2 |
| SUBB A,direct | Subtract direct byte from Acc with borrow | 2 | 3 |
| SUBB A,@Ri | Subtract indirect RAM from Acc with borrow | 1 | 3 |

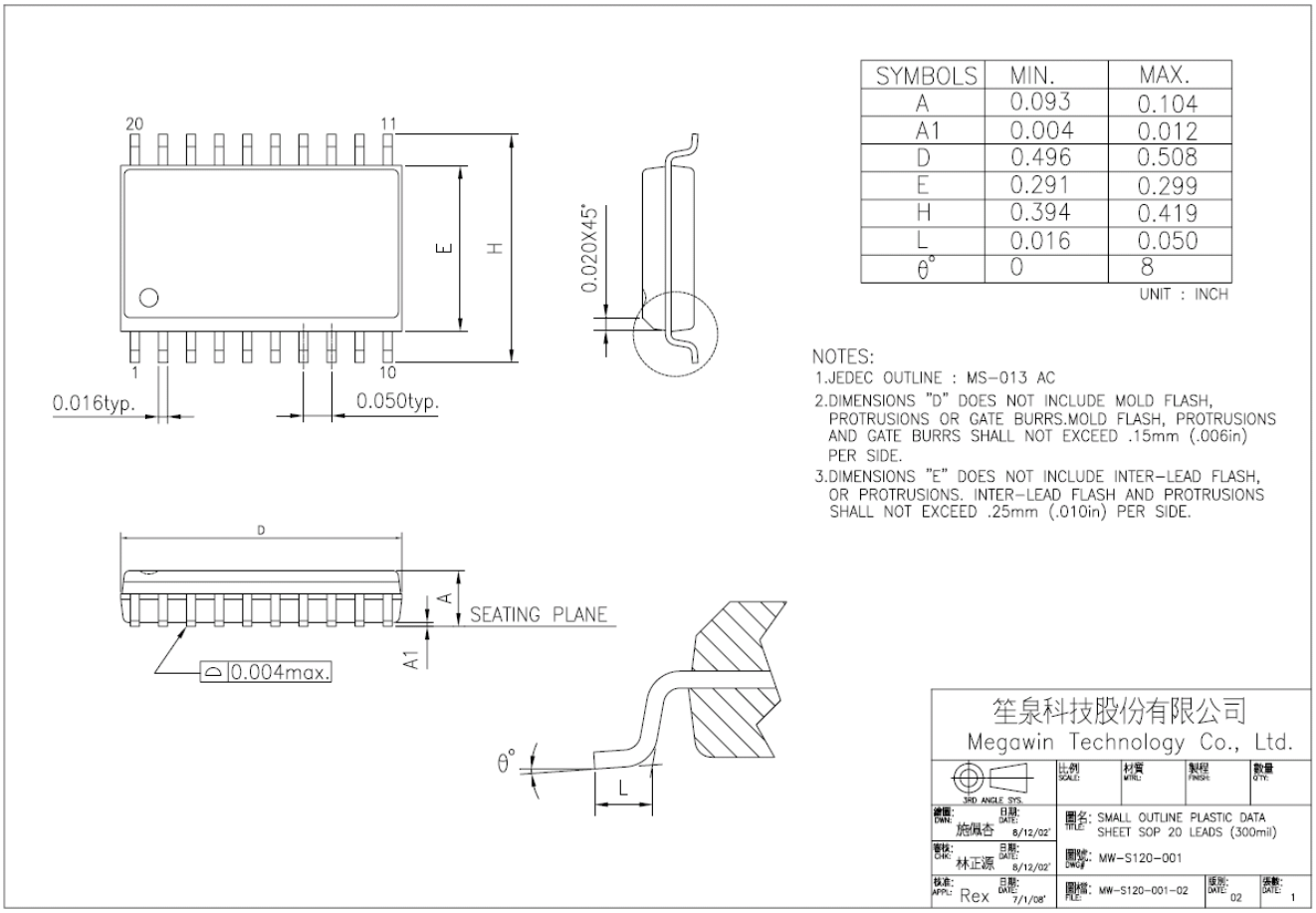
| | | | |
|---|---|---|---|
| SUBB A,#data | Subtract immediate data from Acc with borrow | 2 | 2 |
| INC A | Increment Acc | 1 | 2 |
| INC Rn | Increment register | 1 | 3 |
| INC direct | Increment direct byte | 2 | 4 |
| INC @Ri | Increment indirect RAM | 1 | 4 |
| DEC A | Decrement Acc | 1 | 2 |
| DEC Rn | Decrement register | 1 | 3 |
| DEC direct | Decrement direct byte | 2 | 4 |
| DEC @Ri | Decrement indirect RAM | 1 | 4 |
| INC DPTR | Increment DPTR | 1 | 1 |
| MUL AB | Multiply A and B | 1 | 4 |
| DIV AB | Divide A by B | 1 | 5 |
| DA A | Decimal Adjust Acc | 1 | 4 |
| ***LOGIC OPERATION*** | | | |
| ANL A,Rn | AND register to Acc | 1 | 2 |
| ANL A,direct | AND direct byte to Acc | 2 | 3 |
| ANL A,@Ri | AND indirect RAM to Acc | 1 | 3 |
| ANL A,#data | AND immediate data to Acc | 2 | 2 |
| ANL direct,A | AND Acc to direct byte | 2 | 4 |
| ANL direct,#data | AND immediate data to direct byte | 3 | 4 |
| ORL A,Rn | OR register to Acc | 1 | 2 |
| ORL A,direct | OR direct byte to Acc | 2 | 3 |
| ORL A,@Ri | OR indirect RAM to Acc | 1 | 3 |
| ORL A,#data | OR immediate data to Acc | 2 | 2 |
| ORL direct,A | OR Acc to direct byte | 2 | 4 |
| ORL direct,#data | OR immediate data to direct byte | 3 | 4 |
| XRL A,Rn | Exclusive-OR register to Acc | 1 | 2 |
| XRL A,direct | Exclusive-OR direct byte to Acc | 2 | 3 |
| XRL A,@Ri | Exclusive-OR indirect RAM to Acc | 1 | 3 |
| XRL A,#data | Exclusive-OR immediate data to Acc | 2 | 2 |
| XRL direct,A | Exclusive-OR Acc to direct byte | 2 | 4 |
| XRL direct,#data | Exclusive-OR immediate data to direct byte | 3 | 4 |
| CLR A | Clear Acc | 1 | 1 |
| CPL A | Complement Acc | 1 | 2 |
| RL A | Rotate Acc Left | 1 | 1 |
| RLC A | Rotate Acc Left through the Carry | 1 | 1 |
| RR A | Rotate Acc Right | 1 | 1 |
| RRC A | Rotate Acc Right through the Carry | 1 | 1 |
| SWAP A | Swap nibbles within the Acc | 1 | 1 |
| ***BOOLEAN VARIABLE MANIPULATION*** | | | |
| CLR C | Clear Carry | 1 | 1 |
| CLR bit | Clear direct bit | 2 | 4 |
| SETB C | Set Carry | 1 | 1 |
| SETB bit | Set direct bit | 2 | 4 |
| CPL C | Complement Carry | 1 | 1 |
| CPL bit | Complement direct bit | 2 | 4 |
| ANL C,bit | AND direct bit to Carry | 2 | 3 |
| ANL C,/bit | AND complement of direct bit to Carry | 2 | 3 |
| ORL C,bit | OR direct bit to Carry | 2 | 3 |
| ORL C,/bit | OR complement of direct bit to Carry | 2 | 3 |

| | | | |
|---|---|---|---|
| MOV C,bit | Move direct bit to Carry | 2 | 3 |
| MOV bit,C | Move Carry to direct bit | 2 | 4 |
| ***BOOLEAN VARIABLE MANIPULATION*** | | | |
| JC rel | Jump if Carry is set | 2 | 3 |
| JNC rel | Jump if Carry not set | 2 | 3 |
| JB bit,rel | Jump if direct bit is set | 3 | 4 |
| JNB bit,rel | Jump if direct bit not set | 3 | 4 |
| JBC bit,rel | Jump if direct bit is set and then clear bit | 3 | 5 |
| ***PROAGRAM BRACHING*** | | | |
| ACALL addr11 | Absolute subroutine call | 2 | 6 |
| LCALL addr16 | Long subroutine call | 3 | 6 |
| RET | Return from subroutine | 1 | 4 |
| RETI | Return from interrupt subroutine | 1 | 4 |
| AJMP addr11 | Absolute jump | 2 | 3 |
| LJMP addr16 | Long jump | 3 | 4 |
| SJMP rel | Short jump | 2 | 3 |
| JMP @A+DPTR | Jump indirect relative to DPTR | 1 | 3 |
| JZ rel | Jump if Acc is zero | 2 | 3 |
| JNZ rel | Jump if Acc not zero | 2 | 3 |
| CJNE A,direct,rel | Compare direct byte to Acc and jump if not equal | 3 | 5 |
| CJNE A,#data,rel | Compare immediate data to Acc and jump if not equal | 3 | 4 |
| CJNE Rn,#data,rel | Compare immediate data to register and jump if not equal | 3 | 4 |
| CJNE @Ri,#data,rel | Compare immediate data to indirect RAM and jump if not equal | 3 | 5 |
| DJNZ Rn,rel | Decrement register and jump if not equal | 2 | 4 |
| DJNZ direct,rel | Decrement direct byte and jump if not equal | 3 | 5 |
| NOP | No Operation | 1 | 1 |

# 24. Package Dimension

## 24.1. SOP-20

Figure 24–1. SOP-20




| SYMBOLS | MIN. | MAX. |
|---|---|---|
| A | 0.093 | 0.104 |
| A1 | 0.004 | 0.012 |
| D | 0.496 | 0.508 |
| E | 0.291 | 0.299 |
| H | 0.394 | 0.419 |
| L | 0.016 | 0.050 |
| θ° | 0 | 8 |

UNIT : INCH

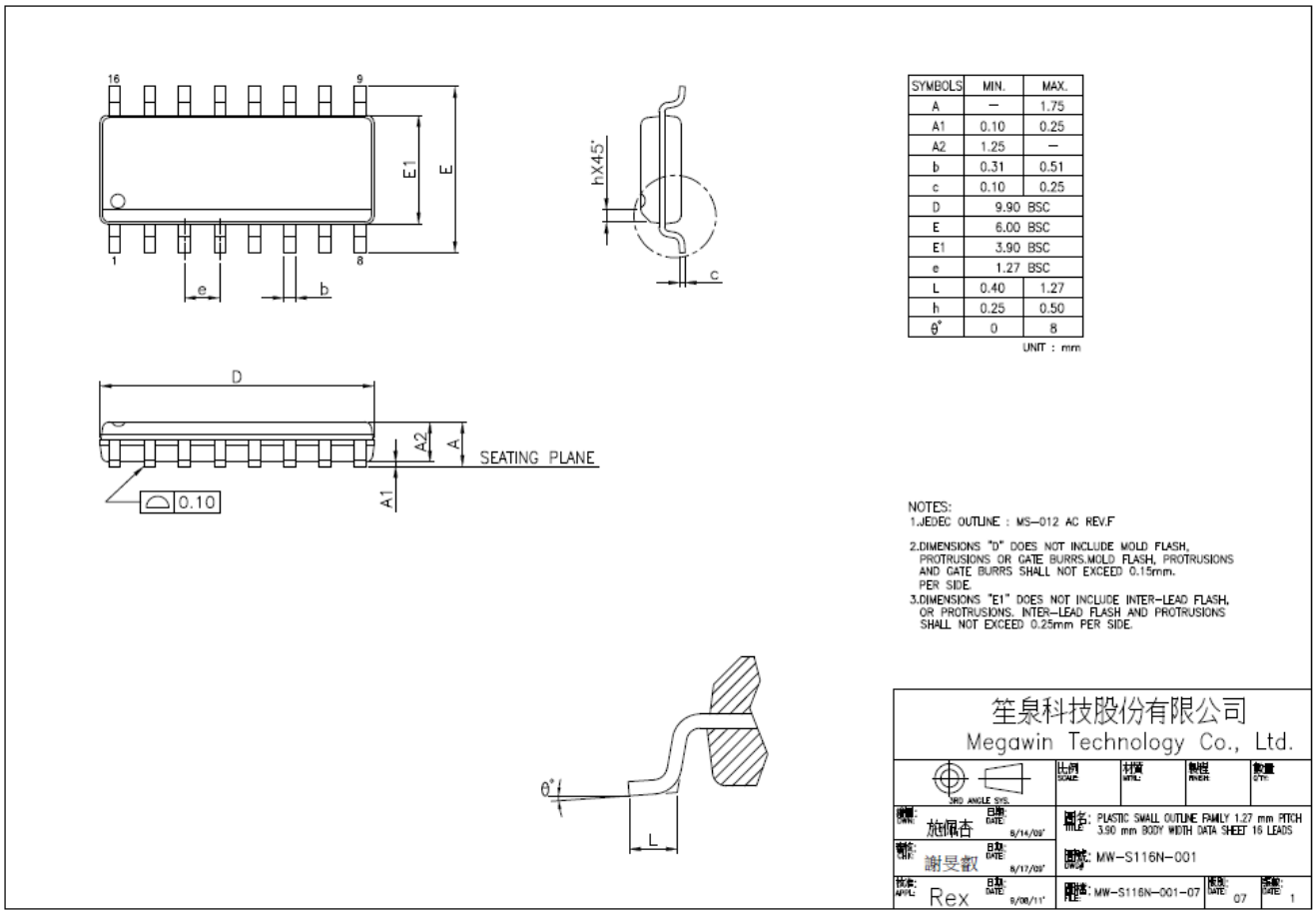
NOTES:
1.JEDEC OUTLINE : MS-013 AC
2.DIMENSIONS "D" DOES NOT INCLUDE MOLD FLASH, PROTRUSIONS OR GATE BURRS.MOLD FLASH, PROTRUSIONS AND GATE BURRS SHALL NOT EXCEED .15mm (.006in) PER SIDE.
3.DIMENSIONS "E" DOES NOT INCLUDE INTER-LEAD FLASH, OR PROTRUSIONS. INTER-LEAD FLASH AND PROTRUSIONS SHALL NOT EXCEED .25mm (.010in) PER SIDE.

笙泉科技股份有限公司
Megawin Technology Co., Ltd.

| | | | |
|---|---|---|---|
| 3RD ANGLE SYS. | 比例 SCALE: | 材質 MTRL: | 製程 FINISH: | 數量 QTY: |
| 繪圖: DWN: 施佩杏 日期: DATE: 8/12/02' | 圖名: TITLE: SMALL OUTLINE PLASTIC DATA SHEET SOP 20 LEADS (300mil) | | | |
| 審核: CHK: 林正源 日期: DATE: 8/12/02' | 圖號: DWG#: MW-S120-001 | | | |
| 核准: APPL: Rex 日期: DATE: 7/1/08' | 圖檔: FILE: MW-S120-001-02 | 版別: DWG 02 | 張數: DATE 1 | |

## 24.2. SOP-16

Figure 24–2. SOP-16




| SYMBOLS | MIN. | MAX. |
|---|---|---|
| A | — | 1.75 |
| A1 | 0.10 | 0.25 |
| A2 | 1.25 | — |
| b | 0.31 | 0.51 |
| c | 0.10 | 0.25 |
| D | 9.90 BSC | |
| E | 6.00 BSC | |
| E1 | 3.90 BSC | |
| e | 1.27 BSC | |
| L | 0.40 | 1.27 |
| h | 0.25 | 0.50 |
| θ° | 0 | 8 |

UNIT : mm

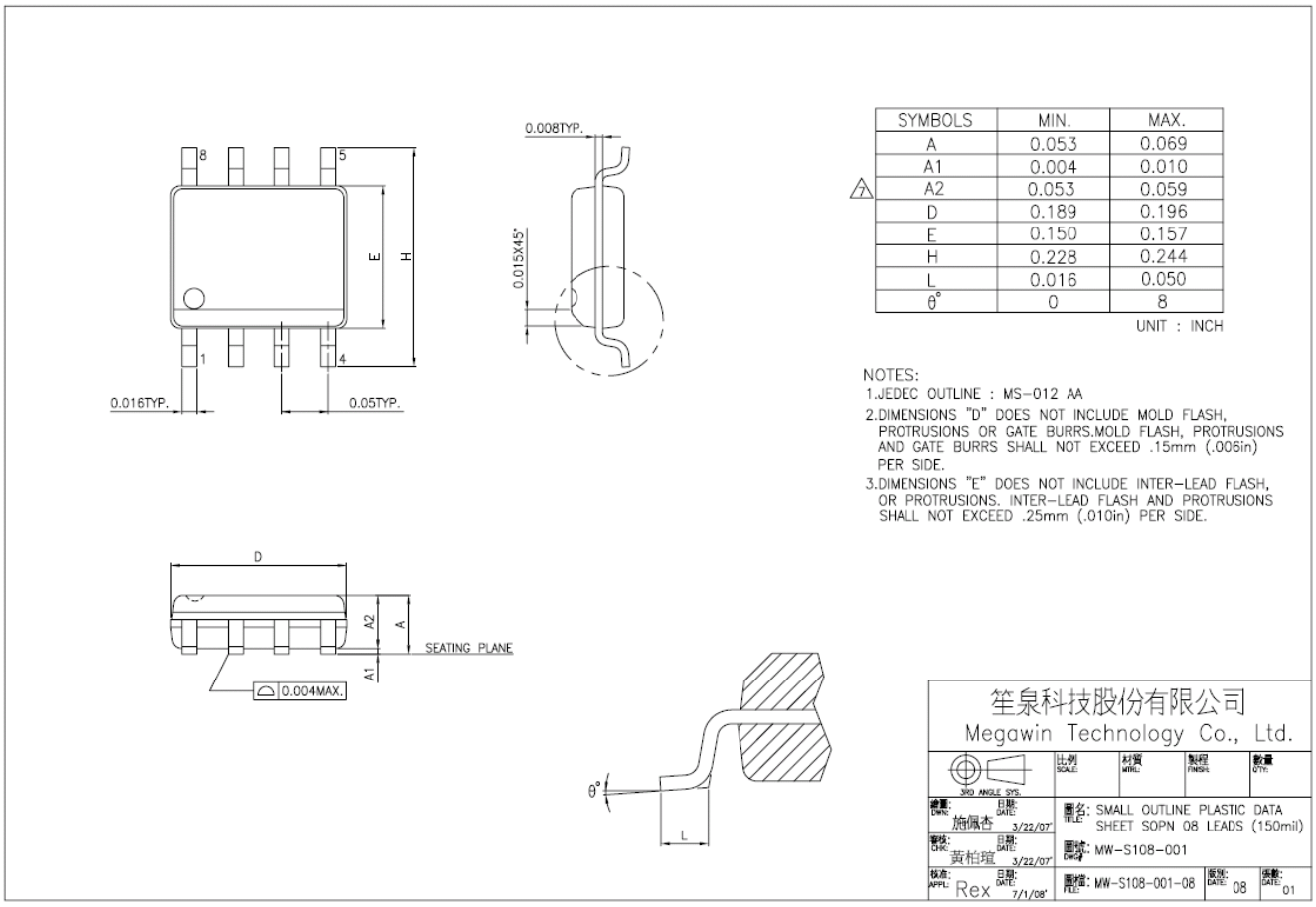
NOTES:
1.JEDEC OUTLINE : MS-012 AC REV.F
2.DIMENSIONS "D" DOES NOT INCLUDE MOLD FLASH, PROTRUSIONS OR GATE BURRS.MOLD FLASH, PROTRUSIONS AND GATE BURRS SHALL NOT EXCEED 0.15mm. PER SIDE.
3.DIMENSIONS "E1" DOES NOT INCLUDE INTER-LEAD FLASH, OR PROTRUSIONS. INTER-LEAD FLASH AND PROTRUSIONS SHALL NOT EXCEED 0.25mm PER SIDE.

笙泉科技股份有限公司
Megawin Technology Co., Ltd.

| | | | | |
|---|---|---|---|---|
| 3RD ANGLE SYS. | 比例 SCALE: | 材質 MTRL: | 粗糙度 FINISH: | 數量 QTY: |
| 繪圖 DWN: 施佩杏 | 日期 DATE: 8/14/09' | 圖名: PLASTIC SMALL OUTLINE FAMILY 1.27 mm PITCH 3.90 mm BODY WIDTH DATA SHEET 16 LEADS | | |
| 審核 CHK: 謝旻叡 | 日期 DATE: 8/17/09' | 圖號 DWG#: MW-S116N-001 | | |
| 核准 APP'D: Rex | 日期 DATE: 9/08/11' | 圖檔 FILE: MW-S116N-001-07 | 版次 DATE: 07 | 張次 DATE: 1 |

## 24.3. SOP-8

Figure 24–3. SOP-8




| SYMBOLS | MIN. | MAX. |
|---|---|---|
| A | 0.053 | 0.069 |
| A1 | 0.004 | 0.010 |
| A2 | 0.053 | 0.059 |
| D | 0.189 | 0.196 |
| E | 0.150 | 0.157 |
| H | 0.228 | 0.244 |
| L | 0.016 | 0.050 |
| θ° | 0 | 8 |

UNIT : INCH

# 25. Revision History

Table 25–1. Revision History

| Rev | Descriptions | Date |
|---|---|---|
| v0.19 | 1. Release preliminary version, v0.19. | 2011/02/25 |
| | | |
| | | |
| V0.20 | 1. Modify to release format. | 2012/01/11 |
| | 2. Merge Section “ISP” & “IAP” and modify Section “Electrical Characteristics” | 2012/01/11 |
| | 3. Add Section “Application Note” | 2012/01/11 |
| | 4. Delete DIP series package. | 2012/01/18 |
| | 5. In Table. 13-1, modify I/O pin-out of SOP8 package | 2012/01/18 |
| | 6. Add package dimension.<br>7. CKCON0.7 AFS bit, default 0 after reset should be IHRCO=24MHz | 2012/01/19<br>2012/09/24 |
| | | |
| A1 | New Datasheet Format | 2013/01/11 |
| | Modify SMOD as SMOD1 in baudrate setting section. | 2013/01/24 |
| A1.2 | Modify MTP write access cycle. | 2013/09/04 |
| A1.3 | Modify IE SFR address error E8h as A8h. | 2013/10/09 |
| A1.4 | Modify SOP-16 package dimension data 300mil to 150mil. | 2014/03/31 |
| A1.5 | Modify SOP-16 package format | 2015/05/27 |

**Disclaimers**

Herein, Megawin stands for "***Megawin Technology Co., Ltd.***"

**Life Support** — This product is not designed for use in medical, life-saving or life-sustaining applications, or systems where malfunction of this product can reasonably be expected to result in personal injury. Customers using or selling this product for use in such applications do so at their own risk and agree to fully indemnify Megawin for any damages resulting from such improper use or sale.

**Right to Make Changes** — Megawin reserves the right to make changes in the products - including circuits, standard cells, and/or software - described or contained herein in order to improve design and/or performance. When the product is in mass production, relevant changes will be communicated via an Engineering Change Notification (ECN).