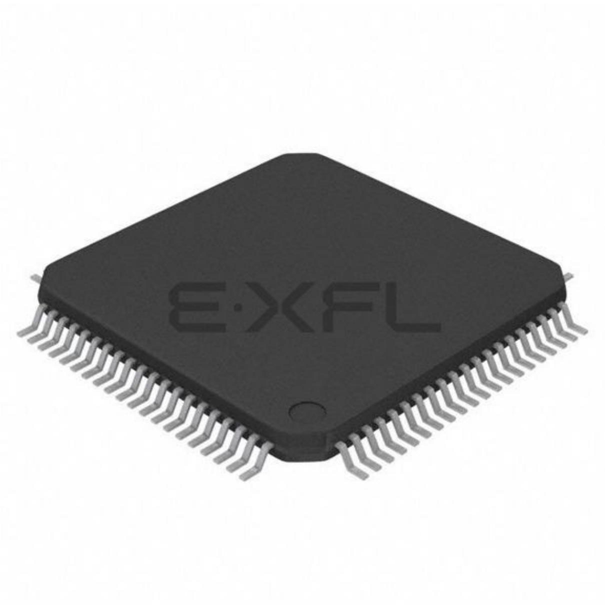


Welcome to E-XFL.COM

What is "Embedded - Microcontrollers"?

"Embedded - Microcontrollers" refer to small, integrated circuits designed to perform specific tasks within larger systems. These microcontrollers are essentially compact computers on a single chip, containing a processor core, memory, and programmable input/output peripherals. They are called "embedded" because they are embedded within electronic devices to control various functions, rather than serving as standalone computers. Microcontrollers are crucial in modern electronics, providing the intelligence and control needed for a wide range of applications.

Applications of "Embedded - Microcontrollers"

Details	
Product Status	Active
Core Processor	PIC
Core Size	8-Bit
Speed	40MHz
Connectivity	EBI/EMI, I²C, SPI, UART/USART
Peripherals	Brown-out Detect/Reset, LVD, POR, PWM, WDT
Number of I/O	68
Program Memory Size	32KB (16K x 16)
Program Memory Type	FLASH
EEPROM Size	1K x 8
RAM Size	2K x 8
Voltage - Supply (Vcc/Vdd)	4.2V ~ 5.5V
Data Converters	A/D 16x10b
Oscillator Type	Internal
Operating Temperature	-40°C ~ 85°C (TA)
Mounting Type	Surface Mount
Package / Case	80-TQFP
Supplier Device Package	80-TQFP (12x12)
Purchase URL	https://www.e-xfl.com/product-detail/microchip-technology/pic18f8520-i-pt

4.9 Data Memory Organization

The data memory is implemented as static RAM. Each register in the data memory has a 12-bit address, allowing up to 4096 bytes of data memory. The data memory map is in turn divided into 16 banks of 256 bytes each. The lower 4 bits of the Bank Select Register (BSR<3:0>) select which bank will be accessed. The upper 4 bits of the BSR are not implemented.

The data memory space contains both Special Function Registers (SFR) and General Purpose Registers (GPR). The SFRs are used for control and status of the controller and peripheral functions, while GPRs are used for data storage and scratch pad operations in the user's application. The SFRs start at the last location of Bank 15 (0FFFh) and extend downwards. Any remaining space beyond the SFRs in the Bank may be implemented as GPRs. GPRs start at the first location of Bank 0 and grow upwards. Any read of an unimplemented location will read as '`0`'s.

PIC18FX520 devices have 2048 bytes of data RAM, extending from Bank 0 to Bank 7 (000h through 7FFh). PIC18FX620 and PIC18FX720 devices have 3840 bytes of data RAM, extending from Bank 0 to Bank 14 (000h through EFFh). The organization of the data memory space for these devices is shown in Figure 4-6 and Figure 4-7.

The entire data memory may be accessed directly or indirectly. Direct addressing may require the use of the BSR register. Indirect addressing requires the use of a File Select Register (FSRn) and a corresponding Indirect File Operand (INDFn). Each FSR holds a 12-bit address value that can be used to access any location in the data memory map without banking.

The instruction set and architecture allow operations across all banks. This may be accomplished by indirect addressing, or by the use of the `MOVFF` instruction. The `MOVFF` instruction is a two-word/two-cycle instruction that moves a value from one register to another.

To ensure that commonly used registers (SFRs and select GPRs) can be accessed in a single cycle, regardless of the current BSR values, an Access Bank is implemented. A segment of Bank 0 and a segment of Bank 15 comprise the Access RAM. **Section 4.10 "Access Bank"** provides a detailed description of the Access RAM.

4.9.1 GENERAL PURPOSE REGISTER FILE

The register file can be accessed either directly or indirectly. Indirect addressing operates using a File Select Register and corresponding Indirect File Operand. The operation of indirect addressing is shown in **Section 4.12 "Indirect Addressing, INDF and FSR Registers"**.

Enhanced MCU devices may have banked memory in the GPR area. GPRs are not initialized by a Power-on Reset and are unchanged on all other Resets.

Data RAM is available for use as General Purpose Registers by all instructions. The top section of Bank 15 (F60h to FFFh) contains SFRs. All other banks of data memory contain GPR registers, starting with Bank 0.

4.9.2 SPECIAL FUNCTION REGISTERS

The Special Function Registers (SFRs) are registers used by the CPU and peripheral modules for controlling the desired operation of the device. These registers are implemented as static RAM. A list of these registers is given in Table 4-2 and Table 4-3.

The SFRs can be classified into two sets: those associated with the "core" function and those related to the peripheral functions. Those registers related to the "core" are described in this section, while those related to the operation of the peripheral features are described in the section of that peripheral feature. The SFRs are typically distributed among the peripherals whose functions they control.

The unused SFR locations are unimplemented and read as '`0`'s. The addresses for the SFRs are listed in Table 4-2.

© 2003-2013 Microchip Technology Inc.

TABLE 4-2: SPECIAL FUNCTION REGISTER MAP

Address	Name	Address	Name	Address	Name	Address	Name
FFFh	TOSU	FDFh	INDF2(3)	FBFh	CCPR1H	F9Fh	IPR1
FFEh	TOSH	FDEh	POSTINC2(3)	FBEh	CCPR1L	F9Eh	PIR1
FFDh	TOSL	FDDh	POSTDEC2(3)	FBDh	CCP1CON	F9Dh	PIE1
FFCh	STKPTR	FDCh	PREINC2(3)	FBCh	CCPR2H	F9Ch	MEMCON(2)
FFBh	PCLATU	FDBh	PLUSW2(3)	FBBh	CCPR2L	F9Bh	—(1)
FFAh	PCLATH	FDAh	FSR2H	FBAh	CCP2CON	F9Ah	TRISJ
FF9h	PCL	FD9h	FSR2L	FB9h	CCPR3H	F99h	TRISH
FF8h	TBLPTRU	FD8h	STATUS	FB8h	CCPR3L	F98h	TRISG
FF7h	TBLPTRH	FD7h	TMR0H	FB7h	CCP3CON	F97h	TRISF
FF6h	TBLPTRL	FD6h	TMR0L	FB6h	—(1)	F96h	TRISE
FF5h	TABLAT	FD5h	T0CON	FB5h	CVRCON	F95h	TRISD
FF4h	PRODH	FD4h	—(1)	FB4h	CMCON	F94h	TRISC
FF3h	PRODL	FD3h	OSCCON	FB3h	TMR3H	F93h	TRISB
FF2h	INTCON	FD2h	LVDCON	FB2h	TMR3L	F92h	TRISA
FF1h	INTCON2	FD1h	WDTCON	FB1h	T3CON	F91h	LATJ
FF0h	INTCON3	FD0h	RCON	FB0h	PSPCON	F90h	LATH
FEFh	INDF0(3)	FCFh	TMR1H	FAFh	SPBRG1	F8Fh	LATG
FEEh	POSTINC0(3)	FCEh	TMR1L	FAEh	RCREG1	F8Eh	LATF
FEDh	POSTDEC0(3)	FCDh	T1CON	FADh	TXREG1	F8Dh	LATE
FECh	PREINC0(3)	FCCh	TMR2	FACh	TXSTA1	F8Ch	LATD
FEBh	PLUSW0(3)	FCBh	PR2	FABh	RCSTA1	F8Bh	LATC
FEAh	FSR0H	FCAh	T2CON	FAAh	EEADRH	F8Ah	LATB
FE9h	FSR0L	FC9h	SSPBUF	FA9h	EEADR	F89h	LATA
FE8h	WREG	FC8h	SSPADD	FA8h	EEDATA	F88h	PORTJ
FE7h	INDF1(3)	FC7h	SSPSTAT	FA7h	EECON2	F87h	PORTH
FE6h	POSTINC1(3)	FC6h	SSPCON1	FA6h	EECON1	F86h	PORTG
FE5h	POSTDEC1(3)	FC5h	SSPCON2	FA5h	IPR3	F85h	PORTF
FE4h	PREINC1(3)	FC4h	ADRESH	FA4h	PIR3	F84h	PORTE
FE3h	PLUSW1(3)	FC3h	ADRESL	FA3h	PIE3	F83h	PORTD
FE2h	FSR1H	FC2h	ADCON0	FA2h	IPR2	F82h	PORTC
FE1h	FSR1L	FC1h	ADCON1	FA1h	PIR2	F81h	PORTB
FE0h	BSR	FC0h	ADCON2	FA0h	PIE2	F80h	PORTA

Note 1: Unimplemented registers are read as '0'.

2: This register is unused on PIC18F6X20 devices. Always maintain this register clear.

3: This is not a physical register.

© 2003-2013 Microchip Technology Inc.

REGISTER 7-1: EECON1 REGISTER (ADDRESS FA6h)

R/W-x	R/W-x	U-0	R/W-0	R/W-x	R/W-0	R/S-0	R/S-0
EEPGD	CFGS	—	FREE	WRERR	WREN	WR	RD
bit 7							bit 0

bit 7 **EEPGD:** Flash Program/Data EEPROM Memory Select bit
1 = Access Flash program memory
0 = Access data EEPROM memory

bit 6 **CFGS:** Flash Program/Data EEPROM or Configuration Select bit
1 = Access configuration or calibration registers
0 = Access Flash program or data EEPROM memory

bit 5 **Unimplemented:** Read as ‘0’

bit 4 **FREE:** Flash Row Erase Enable bit
1 = Erase the program memory row addressed by TBLPTR on the next WR command (cleared by completion of erase operation)
0 = Perform write only

bit 3 **WRERR:** Flash Program/Data EEPROM Error Flag bit
1 = A write operation is prematurely terminated (any $\overline{\text{MCLR}}$ or any WDT Reset during self-timed programming in normal operation)
0 = The write operation completed

Note: When a WRERR occurs, the EEPGD or FREE bits are not cleared. This allows tracing of the error condition.

bit 2 **WREN:** Flash Program/Data EEPROM Write Enable bit
1 = Allows write cycles to Flash program/data EEPROM
0 = Inhibits write cycles to Flash program/data EEPROM

bit 1 **WR:** Write Control bit
1 = Initiates a data EEPROM erase/write cycle, or a program memory erase cycle or write cycle. (The operation is self-timed and the bit is cleared by hardware once write is complete. The WR bit can only be set (not cleared) in software.)
0 = Write cycle to the EEPROM is complete

bit 0 **RD:** Read Control bit
1 = Initiates an EEPROM read. (Read takes one cycle. RD is cleared in hardware. The RD bit can only be set (not cleared) in software. RD bit cannot be set when EEPGD = 1.)
0 = Does not initiate an EEPROM read

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as ‘0’	
- n = Value at POR	‘1’ = Bit is set	‘0’ = Bit is cleared	x = Bit is unknown

© 2003-2013 Microchip Technology Inc.

Example 8-3 shows the sequence to do a 16 x 16 unsigned multiply. Equation 8-1 shows the algorithm that is used. The 32-bit result is stored in four registers, RES3:RES0.

EQUATION 8-1: 16 x 16 UNSIGNED MULTIPLICATION ALGORITHM

$$\begin{aligned}\text{RES3:RES0} &= \text{ARG1H:ARG1L} \bullet \text{ARG2H:ARG2L}\\ &= (\text{ARG1H} \bullet \text{ARG2H} \bullet 2^{16}) +\\ &\quad (\text{ARG1H} \bullet \text{ARG2L} \bullet 2^{8}) +\\ &\quad (\text{ARG1L} \bullet \text{ARG2H} \bullet 2^{8}) +\\ &\quad (\text{ARG1L} \bullet \text{ARG2L})\end{aligned}$$

EXAMPLE 8-3: 16 x 16 UNSIGNED MULTIPLY ROUTINE

```
    MOVF    ARG1L, W
    MULWF   ARG2L          ; ARG1L * ARG2L ->
                           ; PRODH:PRODL
    MOVFF   PRODH, RES1    ;
    MOVFF   PRODL, RES0    ;
;
    MOVF    ARG1H, W
    MULWF   ARG2H          ; ARG1H * ARG2H ->
                           ; PRODH:PRODL
    MOVFF   PRODH, RES3    ;
    MOVFF   PRODL, RES2    ;
;
    MOVF    ARG1L, W
    MULWF   ARG2H          ; ARG1L * ARG2H ->
                           ; PRODH:PRODL
    MOVF    PRODL,  W      ;
    ADDWF   RES1, F        ; Add cross
    MOVF    PRODH, W       ; products
    ADDWFC  RES2, F        ;
    CLRF    WREG           ;
    ADDWFC  RES3, F        ;
;
    MOVF    ARG1H, W       ;
    MULWF   ARG2L          ; ARG1H * ARG2L ->
                           ; PRODH:PRODL
    MOVF    PRODL, W       ;
    ADDWF   RES1, F        ; Add cross
    MOVF    PRODH, W       ; products
    ADDWFC  RES2, F        ;
    CLRF    WREG           ;
    ADDWFC  RES3, F        ;
```

Example 8-4 shows the sequence to do a 16 x 16 signed multiply. Equation 8-2 shows the algorithm used. The 32-bit result is stored in four registers, RES3:RES0. To account for the sign bits of the arguments, each argument pairs' Most Significant bit (MSb) is tested and the appropriate subtractions are done.

EQUATION 8-2: 16 x 16 SIGNED MULTIPLICATION ALGORITHM

$$\begin{aligned}&\text{RES3:RES0}\\ &= \text{ARG1H:ARG1L} \bullet \text{ARG2H:ARG2L}\\ &= (\text{ARG1H} \bullet \text{ARG2H} \bullet 2^{16}) +\\ &\quad (\text{ARG1H} \bullet \text{ARG2L} \bullet 2^{8}) +\\ &\quad (\text{ARG1L} \bullet \text{ARG2H} \bullet 2^{8}) +\\ &\quad (\text{ARG1L} \bullet \text{ARG2L}) +\\ &\quad (-1 \bullet \text{ARG2H<7>} \bullet \text{ARG1H:ARG1L} \bullet 2^{16}) +\\ &\quad (-1 \bullet \text{ARG1H<7>} \bullet \text{ARG2H:ARG2L} \bullet 2^{16})\end{aligned}$$

EXAMPLE 8-4: 16 x 16 SIGNED MULTIPLY ROUTINE

```
    MOVF    ARG1L, W
    MULWF   ARG2L          ; ARG1L * ARG2L ->
                           ; PRODH:PRODL
    MOVFF   PRODH, RES1    ;
    MOVFF   PRODL, RES0    ;
;
    MOVF    ARG1H, W
    MULWF   ARG2H          ; ARG1H * ARG2H ->
                           ; PRODH:PRODL
    MOVFF   PRODH, RES3    ;
    MOVFF   PRODL, RES2    ;
;
    MOVF    ARG1L, W
    MULWF   ARG2H          ; ARG1L * ARG2H ->
                           ; PRODH:PRODL
    MOVF    PRODL, W       ;
    ADDWF   RES1, F        ; Add cross
    MOVF    PRODH, W       ; products
    ADDWFC  RES2, F        ;
    CLRF    WREG           ;
    ADDWFC  RES3, F        ;
;
    MOVF    ARG1H, W       ;
    MULWF   ARG2L          ; ARG1H * ARG2L ->
                           ; PRODH:PRODL
    MOVF    PRODL, W       ;
    ADDWF   RES1, F        ; Add cross
    MOVF    PRODH, W       ; products
    ADDWFC  RES2, F        ;
    CLRF    WREG           ;
    ADDWFC  RES3, F        ;
;
    BTFSS   ARG2H, 7       ; ARG2H:ARG2L neg?
    BRA     SIGN_ARG1      ; no, check ARG1
    MOVF    ARG1L, W       ;
    SUBWF   RES2           ;
    MOVF    ARG1H, W       ;
    SUBWFB  RES3
;
SIGN_ARG1
    BTFSS   ARG1H, 7       ; ARG1H:ARG1L neg?
    BRA     CONT_CODE      ; no, done
    MOVF    ARG2L, W       ;
    SUBWF   RES2           ;
    MOVF    ARG2H, W       ;
    SUBWFB  RES3
;
CONT_CODE
      :
```


© 2003-2013 Microchip Technology Inc.

TABLE 10-11: PORTF FUNCTIONS

Name	Bit#	Buffer Type	Function
RF0/AN5	bit 0	ST	Input/output port pin or analog input.
RF1/AN6/C2OUT	bit 1	ST	Input/output port pin, analog input or comparator 2 output.
RF2/AN7/C1OUT	bit 2	ST	Input/output port pin, analog input or comparator 1 output.
RF3/AN8	bit 3	ST	Input/output port pin or analog input/comparator input.
RF4/AN9	bit 4	ST	Input/output port pin or analog input/comparator input.
RF5/AN10/CVREF	bit 5	ST	Input/output port pin, analog input/comparator input or comparator reference output.
RF6/AN11	bit 6	ST	Input/output port pin or analog input/comparator input.
RF7/$\overline{SS}$	bit 7	ST/TTL	Input/output port pin or slave select pin for synchronous serial port.

Legend: ST = Schmitt Trigger input, TTL = TTL input

TABLE 10-12: SUMMARY OF REGISTERS ASSOCIATED WITH PORTF

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on POR, BOR	Value on all other Resets
TRISF	PORTF Data Direction Control Register								1111 1111	1111 1111
PORTF	Read PORTF pin/Write PORTF Data Latch								xxxx xxxx	uuuu uuuu
LATF	Read PORTF Data Latch/Write PORTF Data Latch								0000 0000	uuuu uuuu
ADCON1	—	—	VCFG1	VCFG0	PCFG3	PCFG2	PCFG1	PCFG0	--00 0000	--00 0000
CMCON	C2OUT	C1OUT	C2INV	C1INV	CIS	CM2	CM1	CM0	0000 0000	0000 0000
CVRCON	CVREN	CVROE	CVRR	CVRSS	CVR3	CVR2	CVR1	CVR0	0000 0000	0000 0000

Legend: x = unknown, u = unchanged. Shaded cells are not used by PORTF.

TABLE 10-13: PORTG FUNCTIONS

Name	Bit#	Buffer Type	Function
RG0/CCP3	bit 0	ST	Input/output port pin or Capture3 input/Compare3 output/PWM3 output.
RG1/TX2/CK2	bit 1	ST	Input/output port pin, addressable USART2 asynchronous transmit or addressable USART2 synchronous clock.
RG2/RX2/DT2	bit 2	ST	Input/output port pin, addressable USART2 asynchronous receive or addressable USART2 synchronous data.
RG3/CCP4	bit 3	ST	Input/output port pin or Capture4 input/Compare4 output/PWM4 output.
RG4/CCP5	bit 4	ST	Input/output port pin or Capture5 input/Compare5 output/PWM5 output.

Legend: ST = Schmitt Trigger input

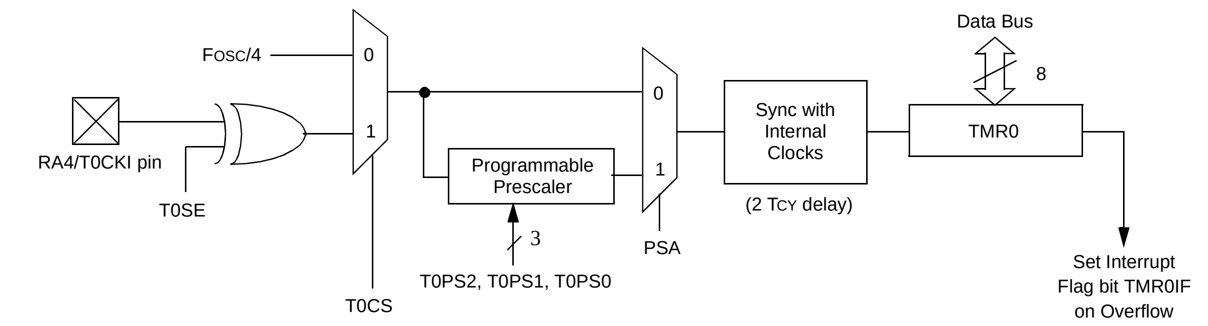
TABLE 10-14: SUMMARY OF REGISTERS ASSOCIATED WITH PORTG

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on POR, BOR	Value on all other Resets
PORTG	—	—	—	Read PORTF pin/Write PORTF Data Latch					`---x xxxx`	`---u uuuu`
LATG	—	—	—	LATG Data Output Register					`---x xxxx`	`---u uuuu`
TRISG	—	—	—	Data Direction Control Register for PORTG					`---1 1111`	`---1 1111`

Legend: `x` = unknown, `u` = unchanged

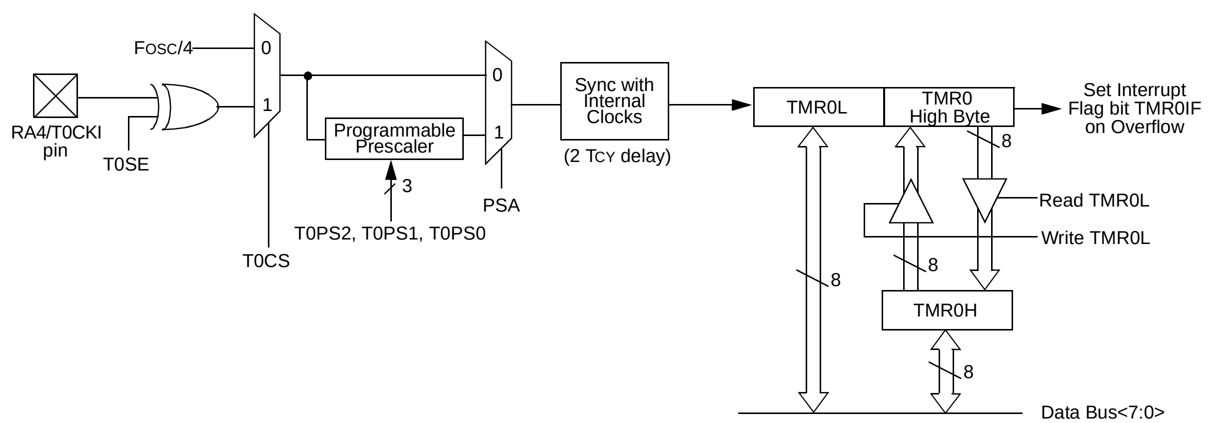
© 2003-2013 Microchip Technology Inc.

FIGURE 11-1: TIMER0 BLOCK DIAGRAM IN 8-BIT MODE



Note: Upon Reset, Timer0 is enabled in 8-bit mode with clock input from T0CKI max. prescale.

FIGURE 11-2: TIMER0 BLOCK DIAGRAM IN 16-BIT MODE



Note: Upon Reset, Timer0 is enabled in 8-bit mode with clock input from T0CKI max. prescale.

© 2003-2013 Microchip Technology Inc.

FIGURE 17-27: BUS COLLISION DURING START CONDITION (SCL = 0)

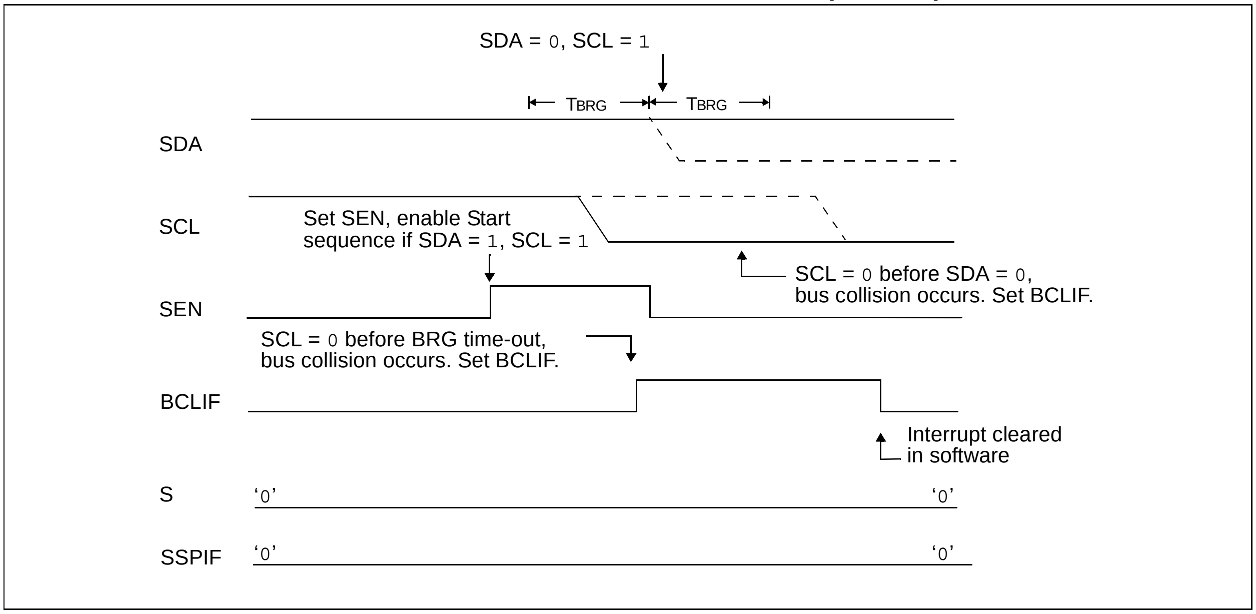
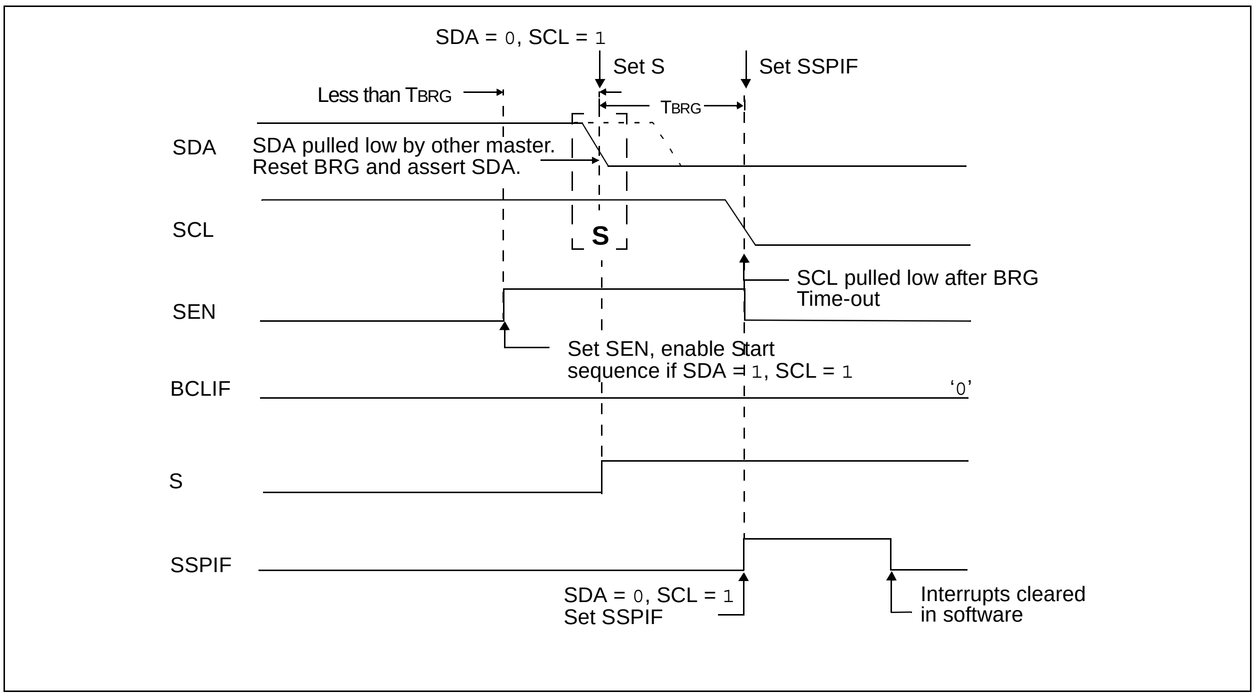


FIGURE 17-28: BRG RESET DUE TO SDA ARBITRATION DURING START CONDITION



REGISTER 19-2: ADCON1 REGISTER

U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	—	VCFG1	VCFG0	PCFG3	PCFG2	PCFG1	PCFG0
bit 7							bit 0

bit 7-6 **Unimplemented:** Read as '0'

bit 5-4 **VCFG1:VCFG0:** Voltage Reference Configuration bits:

VCFG1 VCFG0	A/D VREF+	A/D VREF-
00	AVDD	AVSS
01	External VREF+	AVSS
10	AVDD	External VREF-
11	External VREF+	External VREF-

bit 3-0 **PCFG3:PCFG0:** A/D Port Configuration Control bits:

PCFG3 PCFG0	AN15	AN14	AN13	AN12	AN11	AN10	AN9	AN8	AN7	AN6	AN5	AN4	AN3	AN2	AN1	AN0
0000	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A
0001	D	D	A	A	A	A	A	A	A	A	A	A	A	A	A	A
0010	D	D	D	A	A	A	A	A	A	A	A	A	A	A	A	A
0011	D	D	D	D	A	A	A	A	A	A	A	A	A	A	A	A
0100	D	D	D	D	D	A	A	A	A	A	A	A	A	A	A	A
0101	D	D	D	D	D	D	A	A	A	A	A	A	A	A	A	A
0110	D	D	D	D	D	D	D	A	A	A	A	A	A	A	A	A
0111	D	D	D	D	D	D	D	D	A	A	A	A	A	A	A	A
1000	D	D	D	D	D	D	D	D	D	A	A	A	A	A	A	A
1001	D	D	D	D	D	D	D	D	D	D	A	A	A	A	A	A
1010	D	D	D	D	D	D	D	D	D	D	D	A	A	A	A	A
1011	D	D	D	D	D	D	D	D	D	D	D	D	A	A	A	A
1100	D	D	D	D	D	D	D	D	D	D	D	D	D	A	A	A
1101	D	D	D	D	D	D	D	D	D	D	D	D	D	D	A	A
1110	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	A
1111	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D

A = Analog input D = Digital I/O

Note: Shaded cells indicate A/D channels available only on PIC18F8X20 devices.

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
- n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

 © 2003-2013 Microchip Technology Inc.

22.2 Operation

Depending on the power source for the device voltage, the voltage normally decreases relatively slowly. This means that the LVD module does not need to be constantly operating. To decrease the current requirements, the LVD circuitry only needs to be enabled for short periods, where the voltage is checked. After doing the check, the LVD module may be disabled.

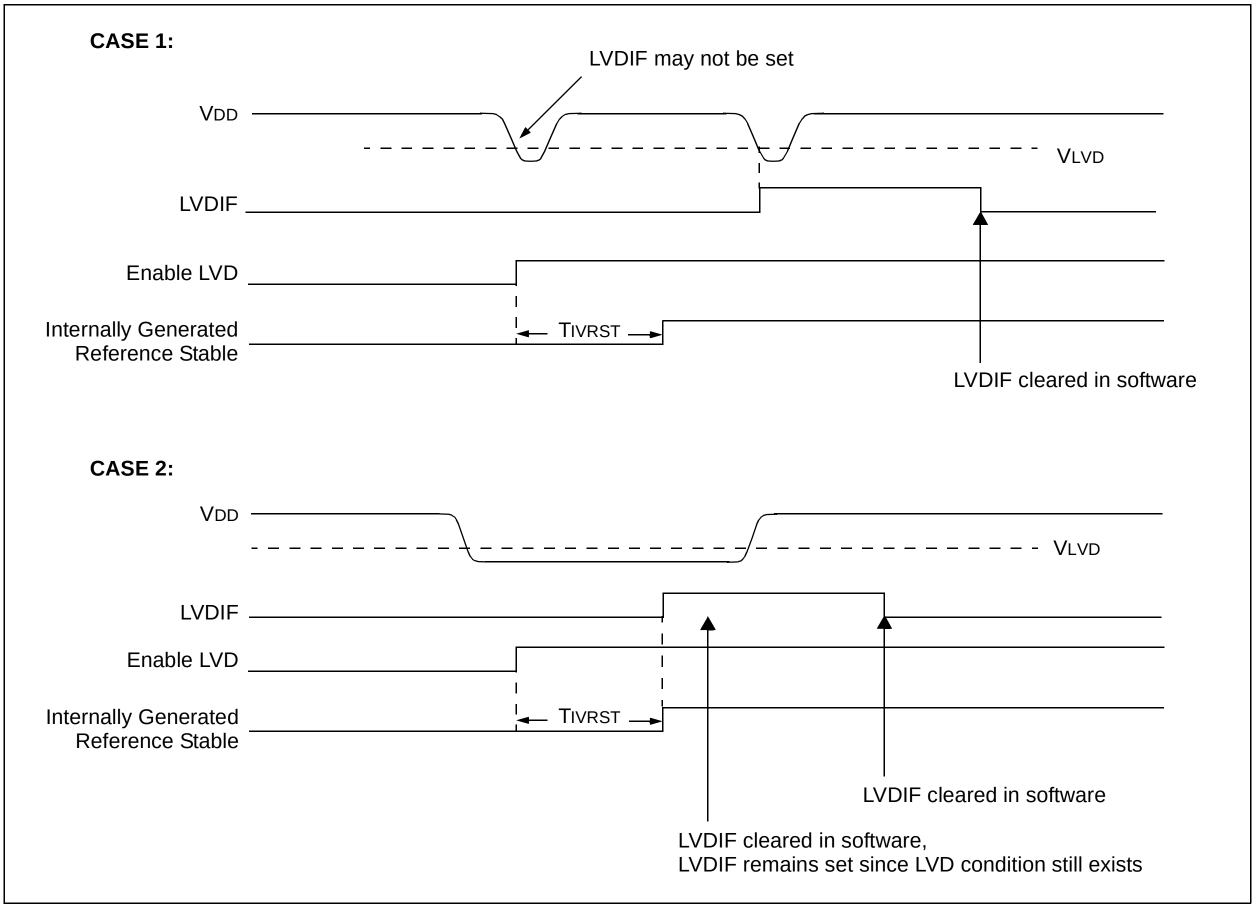
Each time that the LVD module is enabled, the circuitry requires some time to stabilize. After the circuitry has stabilized, all status flags may be cleared. The module will then indicate the proper state of the system.

The following steps are needed to set up the LVD module:

1. Write the value to the LVDL3:LVDL0 bits (LVDCON register), which selects the desired LVD trip point.
2. Ensure that LVD interrupts are disabled (the LVDIE bit is cleared or the GIE bit is cleared).
3. Enable the LVD module (set the LVDEN bit in the LVDCON register).
4. Wait for the LVD module to stabilize (the IRVST bit to become set).
5. Clear the LVD interrupt flag, which may have falsely become set, until the LVD module has stabilized (clear the LVDIF bit).
6. Enable the LVD interrupt (set the LVDIE and the GIE bits).

Figure 22-4 shows typical waveforms that the LVD module may be used to detect.

FIGURE 22-4: LOW-VOLTAGE DETECT WAVEFORMS



© 2003-2013 Microchip Technology Inc.

REGISTER 23-4: CONFIG3L: CONFIGURATION REGISTER 3 LOW (BYTE ADDRESS 300004h)(1)

R/P-1	U-0	U-0	U-0	U-0	U-0	R/P-1	R/P-1
WAIT	—	—	—	—	—	PM1	PM0
bit 7							bit 0

bit 7 **WAIT:** External Bus Data Wait Enable bit
1 = Wait selections unavailable for table reads and table writes
0 = Wait selections for table reads and table writes are determined by the WAIT1:WAIT0 bits (MEMCOM<5:4>)

bit 6-2 **Unimplemented:** Read as '0'

bit 1-0 **PM1:PM0:** Processor Mode Select bits
11 = Microcontroller mode
10 = Microprocessor mode
01 = Microprocessor with Boot Block mode
00 = Extended Microcontroller mode

Note 1: This register is unimplemented in PIC18F6X20 devices; maintain these bits set.

Legend:		
R = Readable bit	P = Programmable bit	U = Unimplemented bit, read as '0'
- n = Value when device is unprogrammed		u = Unchanged from programmed state

REGISTER 23-5: CONFIG3H: CONFIGURATION REGISTER 3 HIGH (BYTE ADDRESS 300005h)

U-0	U-0	U-0	U-0	U-0	U-0	R/P-1	R/P-1
—	—	—	—	—	—	r(1)	CCP2MX
bit 7							bit 0

bit 7-2 **Unimplemented:** Read as '0'

bit 1 **Reserved:** Read as unknown(1)

bit 0 **CCP2MX:** CCP2 Mux bit
<u>In Microcontroller mode:</u>
1 = CCP2 input/output is multiplexed with RC1
0 = CCP2 input/output is multiplexed with RE7
<u>In Microprocessor, Microprocessor with Boot Block and Extended Microcontroller modes (PIC18F8X20 devices only):</u>
1 = CCP2 input/output is multiplexed with RC1
0 = CCP2 input/output is multiplexed with RB3

Note 1: Unimplemented in PIC18FX620 and PIC18FX720 devices; read as '0'.

Legend:		
R = Readable bit	P = Programmable bit	U = Unimplemented bit, read as '0'
- n = Value when device is unprogrammed		u = Unchanged from programmed state

© 2003-2013 Microchip Technology Inc.

REGISTER 23-12: CONFIG7H: CONFIGURATION REGISTER 7 HIGH (BYTE ADDRESS 30000Dh)

U-0	R/P-1	U-0	U-0	U-0	U-0	U-0	U-0
—	EBTRB	—	—	—	—	—	—
bit 7							bit 0

bit 7 **Unimplemented:** Read as '0'

bit 6 **EBTRB:** Boot Block Table Read Protection bit
For PIC18FX520 devices:
1 = Boot Block (000000-0007FFh) not protected from table reads executed in other blocks
0 = Boot Block (000000-0007FFh) protected from table reads executed in other blocks
For PIC18FX620 and PIC18FX720 devices:
1 = Boot Block (000000-0001FFh) not protected from table reads executed in other blocks
0 = Boot Block (000000-0001FFh) protected from table reads executed in other blocks

bit 5-0 **Unimplemented:** Read as '0'

Legend:		
R = Readable bit	P = Programmable bit	U = Unimplemented bit, read as '0'
- n = Value when device is unprogrammed		u = Unchanged from programmed state

REGISTER 23-13: DEVICE ID REGISTER 1 FOR PIC18FXX20 DEVICES (ADDRESS 3FFFFEh)

R	R	R	R	R	R	R	R
DEV2	DEV1	DEV0	REV4	REV3	REV2	REV1	REV0
bit 7							bit 0

bit 7-5 **DEV2:DEV0:** Device ID bits
000 = PIC18F8720
001 = PIC18F6720
010 = PIC18F8620
011 = PIC18F6620

bit 4-0 **REV4:REV0:** Revision ID bits
These bits are used to indicate the device revision.

Legend:		
R = Readable bit	P = Programmable bit	U = Unimplemented bit, read as '0'
- n = Value when device is unprogrammed		u = Unchanged from programmed state

REGISTER 23-14: DEVICE ID REGISTER 2 FOR PIC18FXX20 DEVICES (ADDRESS 3FFFFFh)

R	R	R	R	R	R	R	R
DEV10	DEV9	DEV8	DEV7	DEV6	DEV5	DEV4	DEV3
bit 7							bit 0

bit 7-0 **DEV10:DEV3:** Device ID bits
These bits are used with the DEV2:DEV0 bits in the Device ID Register 1 to identify the part number.

Legend:		
R = Readable bit	P = Programmable bit	U = Unimplemented bit, read as '0'
- n = Value when device is unprogrammed		u = Unchanged from programmed state

BZ — Branch if Zero

Syntax:	[*label*] BZ n
Operands:	$-128 \leq n \leq 127$
Operation:	if Zero bit is '1' (PC) + 2 + 2n $\rightarrow$ PC
Status Affected:	None
Encoding:	`1110` `0000` `nnnn` `nnnn`
Description:	If the Zero bit is '1', then the program will branch. The 2's complement number '2n' is added to the PC. Since the PC will have incremented to fetch the next instruction, the new address will be PC+2+2n. This instruction is then a two-cycle instruction.
Words:	1
Cycles:	1(2)

Q Cycle Activity:

If Jump:

Q1	Q2	Q3	Q4
Decode	Read literal 'n'	Process Data	Write to PC
No operation	No operation	No operation	No operation

If No Jump:

Q1	Q2	Q3	Q4
Decode	Read literal 'n'	Process Data	No operation

Example: `HERE BZ Jump`

Before Instruction

PC = address `(HERE)`

After Instruction

If Zero = 1;
PC = address `(Jump)`
If Zero = 0;
PC = address `(HERE+2)`

CALL — Subroutine Call

Syntax:	[*label*] CALL k [,s]
Operands:	$0 \leq k \leq 1048575$ $s \in [0,1]$
Operation:	(PC) + 4 $\rightarrow$ TOS, k $\rightarrow$ PC<20:1>, if s = 1 (W) $\rightarrow$ WS, (STATUS) $\rightarrow$ STATUSS, (BSR) $\rightarrow$ BSRS
Status Affected:	None

Encoding:

1st word (k<7:0>)	1110	110s	$k_7$kkk	kkkk_0
2nd word(k<19:8>)	1111	k_{19}kkk	kkkk	kkkk_8

Description:	Subroutine call of entire 2-Mbyte memory range. First, return address (PC+ 4) is pushed onto the return stack. If 's' = 1, the W, Status and BSR registers are also pushed into their respective shadow registers, WS, STATUSS and BSRS. If 's' = 0, no update occurs (default). Then, the 20-bit value 'k' is loaded into PC<20:1>. `CALL` is a two-cycle instruction.
Words:	2
Cycles:	2

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read literal 'k'<7:0>	Push PC to stack	Read literal 'k'<19:8>, Write to PC
No operation	No operation	No operation	No operation

Example: `HERE CALL THERE,1`

Before Instruction

PC = address `(HERE)`

After Instruction

PC = address `(THERE)`
TOS = address `(HERE + 4)`
WS = W
BSRS = BSR
STATUSS = STATUS

© 2003-2013 Microchip Technology Inc.

INCFSZ — Increment f, skip if 0

Syntax:	[*label*] INCFSZ f [,d [,a]
Operands:	$0 \le f \le 255$ $d \in [0,1]$ $a \in [0,1]$
Operation:	(f) + 1 → dest, skip if result = 0
Status Affected:	None
Encoding:	`0011` `11da` `ffff` `ffff`
Description:	The contents of register 'f' are incremented. If 'd' is '0', the result is placed in W. If 'd' is '1', the result is placed back in register 'f' (default). If the result is '0', the next instruction which is already fetched is discarded and a NOP is executed instead, making it a two-cycle instruction. If 'a' is '0', the Access Bank will be selected, overriding the BSR value. If 'a' = 1, then the bank will be selected as per the BSR value (default).
Words:	1
Cycles:	1(2) **Note:** 3 cycles if skip and followed by a 2-word instruction.

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register 'f'	Process Data	Write to destination

If skip:

Q1	Q2	Q3	Q4
No operation	No operation	No operation	No operation

If skip and followed by 2-word instruction:

Q1	Q2	Q3	Q4
No operation	No operation	No operation	No operation
No operation	No operation	No operation	No operation

Example:

```
HERE     INCFSZ   CNT, 1, 0
NZERO    :
ZERO     :
```

Before Instruction
- PC = Address (HERE)

After Instruction
- CNT = CNT + 1
- If CNT = 0;
- PC = Address (ZERO)
- If CNT ≠ 0;
- PC = Address (NZERO)

INFSNZ — Increment f, skip if not 0

Syntax:	[*label*] INFSNZ f [,d [,a]
Operands:	$0 \le f \le 255$ $d \in [0,1]$ $a \in [0,1]$
Operation:	(f) + 1 → dest, skip if result ≠ 0
Status Affected:	None
Encoding:	`0100` `10da` `ffff` `ffff`
Description:	The contents of register 'f' are incremented. If 'd' is '0', the result is placed in W. If 'd' is '1', the result is placed back in register 'f' (default). If the result is not '0', the next instruction which is already fetched is discarded and a NOP is executed instead, making it a two-cycle instruction. If 'a' is '0', the Access Bank will be selected, overriding the BSR value. If 'a' = 1, then the bank will be selected as per the BSR value (default).
Words:	1
Cycles:	1(2) **Note:** 3 cycles if skip and followed by a 2-word instruction.

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register 'f'	Process Data	Write to destination

If skip:

Q1	Q2	Q3	Q4
No operation	No operation	No operation	No operation

If skip and followed by 2-word instruction:

Q1	Q2	Q3	Q4
No operation	No operation	No operation	No operation
No operation	No operation	No operation	No operation

Example:

```
HERE     INFSNZ  REG, 1, 0
ZERO
NZERO
```

Before Instruction
- PC = Address (HERE)

After Instruction
- REG = REG + 1
- If REG ≠ 0;
- PC = Address (NZERO)
- If REG = 0;
- PC = Address (ZERO)

MOVFF **Move f to f**

Syntax: [*label*] MOVFF f_s,f_d

Operands: $0 \le f_s \le 4095$
$0 \le f_d \le 4095$

Operation: $(f_s) \rightarrow f_d$

Status Affected: None

Encoding:

1st word (source)	`1100`	`ffff`	`ffff`	`ffff`$_s$
2nd word (destin.)	`1111`	`ffff`	`ffff`	`ffff`$_d$

Description: The contents of source register 'f_s' are moved to destination register 'f_d'. Location of source 'f_s' can be anywhere in the 4096-byte data space (000h to FFFh) and location of destination 'f_d' can also be anywhere from 000h to FFFh. Either source or destination can be W (a useful special situation).
`MOVFF` is particularly useful for transferring a data memory location to a peripheral register (such as the transmit buffer or an I/O port).
The `MOVFF` instruction cannot use the PCL, TOSU, TOSH or TOSL as the destination register.

Words: 2

Cycles: 2 (3)

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register 'f' (src)	Process Data	No operation
Decode	No operation, No dummy read	No operation	Write register 'f' (dest)

Example: `MOVFF REG1, REG2`

Before Instruction
REG1 = 0x33
REG2 = 0x11

After Instruction
REG1 = 0x33,
REG2 = 0x33

MOVLB **Move literal to low nibble in BSR**

Syntax: [*label*] MOVLB k

Operands: $0 \le k \le 255$

Operation: $k \rightarrow BSR$

Status Affected: None

Encoding:

`0000`	`0001`	`kkkk`	`kkkk`

Description: The 8-bit literal 'k' is loaded into the Bank Select Register (BSR).

Words: 1

Cycles: 1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read literal 'k'	Process Data	Write literal 'k' to BSR

Example: `MOVLB 5`

Before Instruction
BSR register = 0x02

After Instruction
BSR register = 0x05

© 2003-2013 Microchip Technology Inc.

25.7 MPLAB SIM Software Simulator

The MPLAB SIM Software Simulator allows code development in a PC-hosted environment by simulating the PIC MCUs and dsPIC® DSCs on an instruction level. On any given instruction, the data areas can be examined or modified and stimuli can be applied from a comprehensive stimulus controller. Registers can be logged to files for further run-time analysis. The trace buffer and logic analyzer display extend the power of the simulator to record and track program execution, actions on I/O, most peripherals and internal registers.

The MPLAB SIM Software Simulator fully supports symbolic debugging using the MPLAB C Compilers, and the MPASM and MPLAB Assemblers. The software simulator offers the flexibility to develop and debug code outside of the hardware laboratory environment, making it an excellent, economical software development tool.

25.8 MPLAB REAL ICE In-Circuit Emulator System

MPLAB REAL ICE In-Circuit Emulator System is Microchip's next generation high-speed emulator for Microchip Flash DSC and MCU devices. It debugs and programs PIC® Flash MCUs and dsPIC® Flash DSCs with the easy-to-use, powerful graphical user interface of the MPLAB Integrated Development Environment (IDE), included with each kit.

The emulator is connected to the design engineer's PC using a high-speed USB 2.0 interface and is connected to the target with either a connector compatible with in-circuit debugger systems (RJ11) or with the new high-speed, noise tolerant, Low-Voltage Differential Signal (LVDS) interconnection (CAT5).

The emulator is field upgradable through future firmware downloads in MPLAB IDE. In upcoming releases of MPLAB IDE, new devices will be supported, and new features will be added. MPLAB REAL ICE offers significant advantages over competitive emulators including low-cost, full-speed emulation, run-time variable watches, trace analysis, complex breakpoints, a ruggedized probe interface and long (up to three meters) interconnection cables.

25.9 MPLAB ICD 3 In-Circuit Debugger System

MPLAB ICD 3 In-Circuit Debugger System is Microchip's most cost effective high-speed hardware debugger/programmer for Microchip Flash Digital Signal Controller (DSC) and microcontroller (MCU) devices. It debugs and programs PIC® Flash microcontrollers and dsPIC® DSCs with the powerful, yet easy-to-use graphical user interface of MPLAB Integrated Development Environment (IDE).

The MPLAB ICD 3 In-Circuit Debugger probe is connected to the design engineer's PC using a high-speed USB 2.0 interface and is connected to the target with a connector compatible with the MPLAB ICD 2 or MPLAB REAL ICE systems (RJ-11). MPLAB ICD 3 supports all MPLAB ICD 2 headers.

25.10 PICkit 3 In-Circuit Debugger/ Programmer and PICkit 3 Debug Express

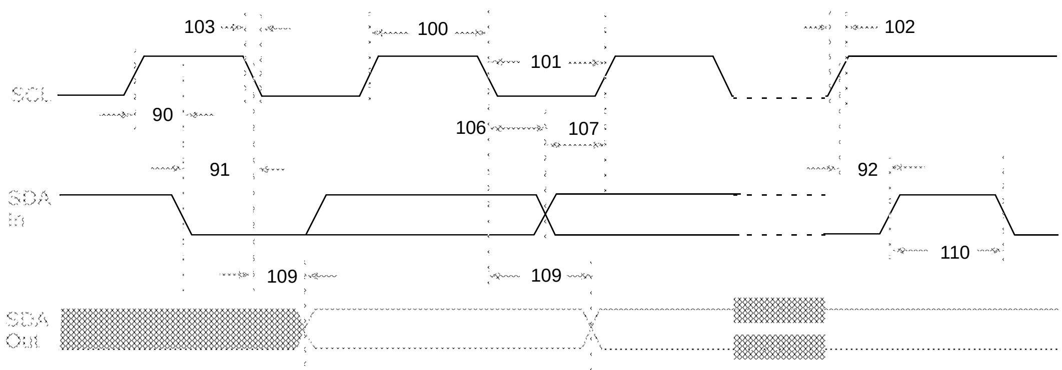
The MPLAB PICkit 3 allows debugging and programming of PIC® and dsPIC® Flash microcontrollers at a most affordable price point using the powerful graphical user interface of the MPLAB Integrated Development Environment (IDE). The MPLAB PICkit 3 is connected to the design engineer's PC using a full speed USB interface and can be connected to the target via an Microchip debug (RJ-11) connector (compatible with MPLAB ICD 3 and MPLAB REAL ICE). The connector uses two device I/O pins and the reset line to implement in-circuit debugging and In-Circuit Serial Programming™.

The PICkit 3 Debug Express include the PICkit 3, demo board and microcontroller, hookup cables and CDROM with user's guide, lessons, tutorial, compiler and MPLAB IDE software.

TABLE 26-19: I^{2}C BUS START/STOP BITS REQUIREMENTS (SLAVE MODE)

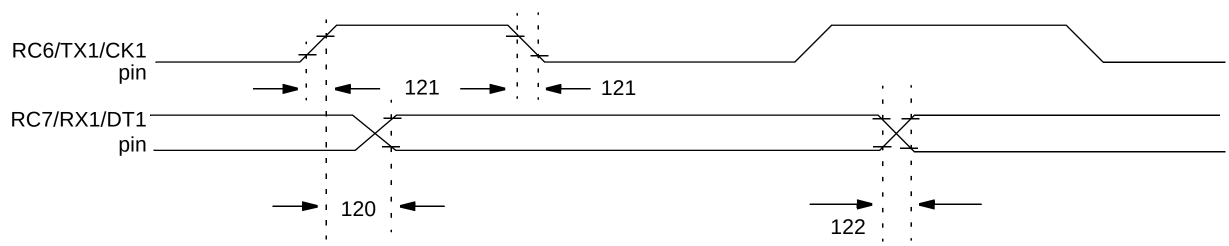
Param No.	Symbol	Characteristic		Min	Max	Units	Conditions
90	TSU:STA	Start Condition	100 kHz mode	4700	—	ns	Only relevant for Repeated Start condition
		Setup Time	400 kHz mode	600	—		
91	THD:STA	Start Condition	100 kHz mode	4000	—	ns	After this period, the first clock pulse is generated
		Hold Time	400 kHz mode	600	—		
92	TSU:STO	Stop Condition	100 kHz mode	4700	—	ns	
		Setup Time	400 kHz mode	600	—		
93	THD:STO	Stop Condition	100 kHz mode	4000	—	ns	
		Hold Time	400 kHz mode	600	—		

FIGURE 26-21: I^{2}C BUS DATA TIMING



Note: Refer to Figure 26-6 for load conditions.

FIGURE 26-24: USART SYNCHRONOUS TRANSMISSION (MASTER/SLAVE) TIMING

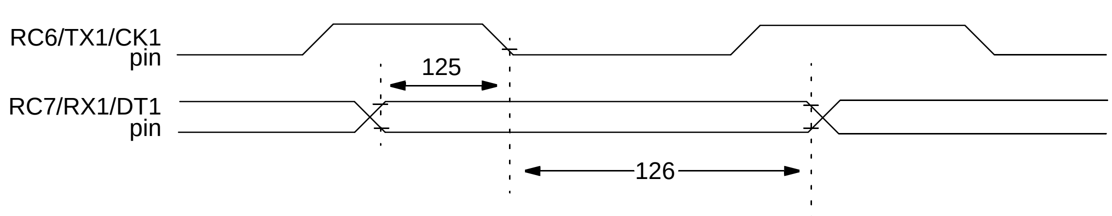


Note: Refer to Figure 26-6 for load conditions.

TABLE 26-23: USART SYNCHRONOUS TRANSMISSION REQUIREMENTS

Param No.	Symbol	Characteristic		Min	Max	Units	Conditions
120	TCKH2DTV	SYNC XMIT (MASTER & SLAVE) Clock High to Data Out Valid	PIC18FXX20	—	40	ns	
			PIC18LFXX20	—	100	ns	VDD = 2.0V
121	TCKRF	Clock Out Rise Time and Fall Time (Master mode)	PIC18FXX20	—	20	ns	
			PIC18LFXX20	—	50	ns	VDD = 2.0V
122	TDTRF	Data Out Rise Time and Fall Time	PIC18FXX20	—	20	ns	
			PIC18LFXX20	—	50	ns	VDD = 2.0V

FIGURE 26-25: USART SYNCHRONOUS RECEIVE (MASTER/SLAVE) TIMING



Note: Refer to Figure 26-6 for load conditions.

TABLE 26-24: USART SYNCHRONOUS RECEIVE REQUIREMENTS

Param No.	Symbol	Characteristic	Min	Max	Units	Conditions
125	TDTV2CKL	SYNC RCV (MASTER & SLAVE) Data Hold before CK ↓ (DT hold time)	10	—	ns	
126	TCKL2DTL	Data Hold after CK ↓ (DT hold time)	15	—	ns	

APPENDIX A: REVISION HISTORY

Revision A (January 2003)

Original data sheet for the PIC18FXX20 family which includes PIC18F6520, PIC18F6620, PIC18F6720, PIC18F8520, PIC18F8620 and PIC18F8720 devices.

This data sheet is based on the previous PIC18FXX20 Data Sheet (DS39580).

Revision B (January 2004)

This revision includes the DC and AC Characteristics Graphs and Tables. The Electrical Specifications in **Section 26.0 "Electrical Characteristics"** have been updated and there have been minor corrections to the data sheet text.

Revision C (November 2011)

This revision updated **Section 28.0 "Packaging Information"**.

APPENDIX B: DEVICE DIFFERENCES

The differences between the devices listed in this data sheet are shown in Table B-1.

TABLE B-1: DEVICE DIFFERENCES

Feature	PIC18F6520	PIC18F6620	PIC18F6720	PIC18F8520	PIC18F8620	PIC18F8720
On-Chip Program Memory (Kbytes)	32	64	128	32	64	128
Data Memory (bytes)	2048	3840	3840	2048	3840	3840
Boot Block (bytes)	2048	512	512	2048	512	512
Timer1 Low-Power Option	Yes	No	No	Yes	No	No
I/O Ports	Ports A, B, C, D, E, F, G	Ports A, B, C, D, E, F, G	Ports A, B, C, D, E, F, G	Ports A, B, C, D, E, F, G, H, J	Ports A, B, C, D, E, F, G, H, J	Ports A, B, C, D, E, F, G, H, J
A/D Channels	12	12	12	16	16	16
External Memory Interface	No	No	No	Yes	Yes	Yes
Maximum Operating Frequency (MHz)	40	25	25	40	25	25
Package Types	64-pin TQFP	64-pin TQFP	64-pin TQFP	80-pin TQFP	80-pin TQFP	80-pin TQFP

READER RESPONSE

It is our intention to provide you with the best documentation possible to ensure successful use of your Microchip product. If you wish to provide your comments on organization, clarity, subject matter, and ways in which our documentation can better serve you, please FAX your comments to the Technical Publications Manager at (480) 792-4150.

Please list the following information, and use this outline to provide us with your comments about this document.

TO: Technical Publications Manager

RE: Reader Response

Total Pages Sent ________

From: Name ________

Company ________

Address ________

City / State / ZIP / Country ________

Telephone: (_______) _________ - _________

FAX: (______) _________ - _________

Application (optional):

Would you like a reply? ___Y ___N

Device: PIC18F6520/8520/6620/8620/6720/8720

Literature Number: DS39609C

Questions:

1. What are the best features of this document?

2. How does this document meet your hardware and software development needs?

3. Do you find the organization of this document easy to follow? If not, why?

4. What additions to the document do you think would enhance the structure and subject?

5. What deletions from the document could be made without affecting the overall usefulness?

6. Is there any incorrect or misleading information (what and where)?

7. How would you improve this document?

© 2003-2013 Microchip Technology Inc.