

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 "<u>Embedded -</u> <u>Microcontrollers</u>"

Details

Product Status	Active
Core Processor	PIC
Core Size	8-Bit
Speed	64MHz
Connectivity	CANbus, I ² C, LINbus, SPI, UART/USART
Peripherals	Brown-out Detect/Reset, DMA, POR, PWM, WDT
Number of I/O	25
Program Memory Size	64KB (32K x 16)
Program Memory Type	FLASH
EEPROM Size	1K x 8
RAM Size	4K x 8
Voltage - Supply (Vcc/Vdd)	1.8V ~ 3.6V
Data Converters	A/D 24x12b; D/A 1x5b
Oscillator Type	Internal
Operating Temperature	-40°C ~ 85°C (TA)
Mounting Type	Surface Mount
Package / Case	28-UQFN Exposed Pad
Supplier Device Package	28-UQFN (6x6)
Purchase URL	https://www.e-xfl.com/product-detail/microchip-technology/pic18lf26k83-i-mx

Email: info@E-XFL.COM

Address: Room A, 16/F, Full Win Commercial Centre, 573 Nathan Road, Mongkok, Hong Kong

4.0 MEMORY ORGANIZATION

There are three types of memory in PIC18 enhanced microcontroller devices:

- Program Flash Memory
- Data RAM
- Data EEPROM

The Program Memory Flash and data RAM share the same bus, while data EEPROM uses a separate bus. This allows for concurrent access of the memory spaces.

Additional detailed information on the operation of the Program Flash Memory and Data EEPROM Memory is provided in Section 13.0 "Nonvolatile Memory (NVM) Control".

4.1 Program Flash Memory Organization

PIC18 microcontrollers implement a 21-bit Program Counter, which is capable of addressing a 2 Mbyte program memory space. Accessing any unimplemented memory will return all '0's (a NOP instruction).

These devices contains the following:

- PIC18(L)F25K83: 32 Kbytes of Flash memory, up to 16,384 single-word instructions
- PIC18(L)F26K83: 64 Kbytes of Flash memory, up to 32,768 single-word instructions

The Reset vector for the device is at address 000000h. PIC18(L)F25/26K83 devices feature a vectored interrupt controller with a dedicated interrupt vector table in the program memory, see **Section 9.0 "Interrupt Controller**".

Note: For memory information on this family of devices, see Table 4-1 and Table 4-3.

4.2 Memory Access Partition (MAP)

Program Flash Memory is partitioned into:

- Application Block
- Boot Block, and
- Storage Area Flash (SAF) Block

4.2.1 APPLICATION BLOCK

Application Block is where the user's program resides by default. Default settings of the Configuration bits (BBEN = 1 and $\overline{SAFEN} = 1$) assign all memory in the Program Flash Memory area to the Application Block. The WRTAPP Configuration bit is used to protect the Application Block.

4.2.2 BOOT BLOCK

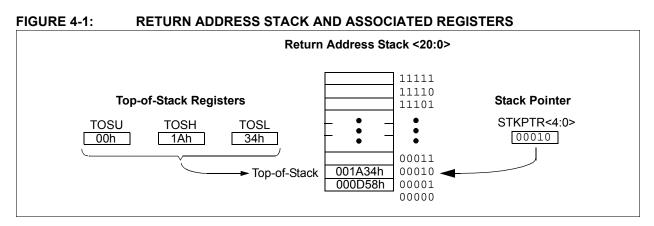
Boot Block is an area in program memory that is ideal for storing bootloader code. Code placed in this area can be executed by the CPU. The Boot Block can be write-protected, independent of the main Application Block. The Boot Block is enabled by the BBEN bit and size is based on the value of the BBSIZE bits of Configuration word (Register 5-7), see Table 5-1 for Boot Block sizes.

The $\overline{\text{WRTB}}$ Configuration bit is used to write-protect the Boot Block.

4.2.3 STORAGE AREA FLASH

Storage Area Flash (SAF) is the area in program memory that can be used as data storage. SAF is enabled by the SAFEN bit of the Configuration word in Register 5-7. If enabled, the code placed in this area cannot be executed by the CPU. The SAF block is placed at the end of memory and spans 256 bytes. The WRTSAF Configuration bit is used to write-protect the Storage Area Flash.

Note: If write-protected locations are written from NVMCON registers, memory is not changed and the WRERR bit defined in Register 13-1 is set.



4.2.5.2 Return Stack Pointer (STKPTR)

The STKPTR register (Register 4-4) contains the Stack Pointer value. The STKOVF (Stack Overflow) Status bit and the STKUNF (Stack Underflow) Status bit can be accessed using the PCON0 register. The value of the Stack Pointer can be 0 through 31. On Reset, the Stack Pointer value will be zero. The user may read and write the Stack Pointer value. This feature can be used by a Real-Time Operating System (RTOS) for stack maintenance. After the PC is pushed onto the stack 32 times (without popping any values off the stack), the STKOVF bit is set. The STKOVF bit is cleared by software or by a POR. The action that takes place when the stack becomes full depends on the state of the STVREN (Stack Overflow Reset Enable) Configuration bit. (Refer to Section 5.1 "Configuration Words" for a description of the device Configuration bits.)

If STVREN is set (default), a Reset will be generated and a Stack Overflow will be indicated by the STKOVF bit when the 32nd push is initiated. This includes CALL and CALLW instructions, as well as stacking the return address during an interrupt response. The STKOVF bit will remain set and the Stack Pointer will be set to zero.

If STVREN is cleared, the STKOVF bit will be set on the 32nd push and the Stack Pointer will remain at 31 but no Reset will occur. Any additional pushes will overwrite the 31st push but the STKPTR will remain at 31.

Setting STKOVF = 1 in software will change the bit, but will not generate a Reset.

The STKUNF bit is set when a stack pop returns a value of zero. The STKUNF bit is cleared by software or by POR. The action that takes place when the stack becomes full depends on the state of the STVREN (Stack Overflow Reset Enable) Configuration bit. (Refer to **Section 5.1 "Configuration Words"** for a description of the device Configuration bits).

If STVREN is set (default) and the stack has been popped enough times to unload the stack, the next pop will return a value of zero to the PC, it will set the STKUNF bit and a Reset will be generated. This condition can be generated by the RETURN, RETLW and RETFIE instructions.

When STVREN = 0, STKUNF will be set but no Reset will occur.

Note: Returning a value of zero to the PC on an underflow has the effect of vectoring the program to the Reset vector, where the stack conditions can be verified and appropriate actions can be taken. This is not the same as a Reset, as the contents of the SFRs are not affected.

4.2.5.3 PUSH and POP Instructions

Since the Top-of-Stack is readable and writable, the ability to push values onto the stack and pull values off the stack without disturbing normal program execution is a desirable feature. The PIC18 instruction set includes two instructions, PUSH and POP, that permit the TOS to be manipulated under software control. TOSU, TOSH and TOSL can be modified to place data or a return address on the stack.

The PUSH instruction places the current PC value onto the stack. This increments the Stack Pointer and loads the current PC value onto the stack.

The POP instruction discards the current TOS by decrementing the Stack Pointer. The previous value pushed onto the stack then becomes the TOS value.

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Register on Page
OSCCON1	_	N	OSC<2:0>			NDIV<	3:0>		94
OSCCON2	_	С	OSC<2:0>			CDIV<	3:0>		95
OSCCON3	CSWHOLD	SOSCPWR	_	ORDY	NOSCR	_	_	_	95
OSCSTAT	EXTOR	HFOR	MFOR	LFOR	SOR	ADOR		PLLR	96
OSCTUNE	_	_			TUN<	5:0>			98
OSCFRQ	—	—	—	—	FRQ<3:0>				97
OSCEN	EXTOEN	HFOEN	MFOEN	LFOEN	SOSCEN	ADOEN	—	—	99

 TABLE 7-3:
 SUMMARY OF REGISTERS ASSOCIATED WITH CLOCK SOURCES

Legend: — = unimplemented location, read as '0'. Shaded cells are not used by clock sources.

9.6 Returning from Interrupt Service Routine (ISR)

The "Return from Interrupt" instruction (RETFIE) is used to mark the end of an ISR.

When RETFIE 1 instruction is executed, the PC is loaded with the saved PC value from the top of the PC stack. Saved context is also restored with the execution of this instruction. Thus, execution returns to the previous state of operation that existed before the interrupt occurred.

When RETFIE 0 instruction is executed, the saved context is not restored back to the registers.

9.7 Interrupt Latency

By assigning each interrupt with a vector address/number (MVECEN = 1), scanning of all interrupts is not necessary to determine the source of the interrupt.

When MVECEN = 1, Vectored interrupt controller requires three clock cycles to vector to the ISR from main routine, thereby removing dependency of interrupt timing on compiled code.

There is a fixed latency of three instruction cycles between the completion of the instruction active when the interrupt occurred and the first instruction of the Interrupt Service Routine. Figure 9-7, Figure 9-8 and Figure 9-9 illustrates the sequence of events when a peripheral interrupt is asserted when the last executed instruction is one-cycle, two-cycle and three-cycle respectively, when MVECEN = 1.

After the Interrupt Flag Status bit is set, the current instruction completes executing. In the first latency cycle, the contents of the PC, STATUS, WREG, BSR, FSR0/1/2, PRODL/H and PCLATH/U registers are context saved and the IVTBASE+ Vector number is calculated. In the second latency cycle, the PC is loaded with the calculated vector table address for the interrupt source and the starting address of the ISR is fetched. In the third latency cycle, the PC is loaded with the ISR address. All the latency cycles are executed as a FNOP instruction.

When MVECEN = 0, Vectored interrupt controller requires two clock cycles to vector to the ISR from main routine. There is a latency of two instruction cycles plus the software latency between the completion of the instruction active when the interrupt occurred and the first instruction of the Interrupt Service Routine.

R-0/0	R-0/0	U-0	U-0	U-0	U-0	U-0	U-0		
STAT<1:0>		—	_	—	—	—	—		
bit 7							bit 0		
Legend:									
HC = Bit is clea	ared by hardwa	ire							
R = Readable	bit	W = Writable I	bit	U = Unimplemented bit, read as '0'					
u = Bit is uncha	anged	x = Bit is unkn	iown	-n/n = Value at POR and BOR/Value at all other Resets					
'1' = Bit is set		'0' = Bit is clea	ared	q = Value depends on condition					

REGISTER 9-2: INTCON1: INTERRUPT CONTROL REGISTER 1

bit 7-6 STAT<1:0>: Interrupt State Status bits

11 = High priority ISR executing, high priority interrupt was received while a low priority ISR was executing

10 = High priority ISR executing, high priority interrupt was received in main routine

01 = Low priority ISR executing, low priority interrupt was received in main routine

00 = Main routine executing

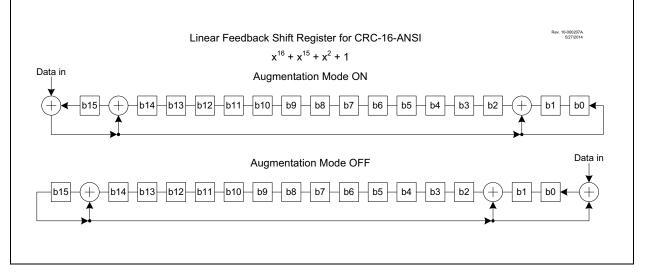
bit 5-0 **Unimplemented**: Read as '0'

14.3 CRC Polynomial Implementation

Any polynomial can be used. The polynomial and accumulator sizes are determined by the PLEN<3:0> bits. For an n-bit accumulator, PLEN = n-1 and the corresponding polynomial is n+1 bits. Therefore the accumulator can be any size up to 16 bits with a corresponding polynomial up to 17 bits. The MSb and LSb of the polynomial are always '1' which is forced by hardware. All polynomial bits between the MSb and LSb are specified by the CRCXOR registers. For example, when using CRC-16-ANSI, the polynomial is defined as $X^{16}+X^{15}+X^2+1$.

The X¹⁶ and X⁰ = 1 terms are the MSb and LSb controlled by hardware. The X¹⁵ and X² terms are specified by setting the corresponding CRCXOR<15:0> bits with the value of '0x8004'. The actual value is '0x8005' because the hardware sets the LSb to 1. However, the LSb of the CRCXORL register is unimplemented and always reads as '0'. Refer to Example 14-1.





14.4 CRC Data Sources

Data can be input to the CRC module in two ways:

- User data using the CRCDAT registers (CRCDATH and CRCDATL)
- Program memory using the Program Memory Scanner

To set the number of bits of data, up to 16 bits, the DLEN bits of CRCCON1 must be set accordingly. Only data bits in CRCDAT registers up to DLEN will be used, other data bits in CRCDAT registers will be ignored.

Data is moved into the CRCSHIFT as an intermediate to calculate the check value located in the CRCACC registers.

The SHIFTM bit is used to determine the bit order of the data being shifted into the accumulator. If SHIFTM is not set, the data will be shifted in MSb first (Big Endian). The value of DLEN will determine the MSb. If SHIFTM bit is set, the data will be shifted into the accumulator in reversed order, LSb first (Little Endian).

The CRC module can be seeded with an initial value by setting the CRCACC<15:0> registers to the appropriate value before beginning the CRC.

14.4.1 CRC FROM USER DATA

To use the CRC module on data input from the user, the user must write the data to the CRCDAT registers. The data from the CRCDAT registers will be latched into the shift registers on any write to the CRCDATL register.

14.4.2 CRC FROM FLASH

To use the CRC module on data located in Program memory, the user can initialize the Program Memory Scanner as defined in **Section 14.8, Scanner Module Overview**.

R/W-1/1	R/W-1/1	R/W-1/1	R/W-1/1	R/W-1/1	R/W-1/1	R/W-1/1				
HADR<15:8> ^(1, 2)										
						bit 0				
oit	W = Writable	bit	U = Unimplemented bit, read as '0'							
anged	x = Bit is unkr	nown	-n/n = Value at POR and BOR/Value at all other Resets							
	'0' = Bit is clea	ared								
	pit	pit W = Writable anged x = Bit is unkr	HADR<	HADR < 15:8 > (1, 2) bit W = Writable bit U = Unimpler anged x = Bit is unknown -n/n = Value a	HADR < 15:8 > (1, 2) bit W = Writable bit U = Unimplemented bit, reac anged x = Bit is unknown -n/n = Value at POR and BO	HADR<15:8>(1, 2)DitW = Writable bitU = Unimplemented bit, read as '0'angedx = Bit is unknown-n/n = Value at POR and BOR/Value at all c				

REGISTER 14-16: SCANHADRH: SCAN HIGH ADDRESS HIGH BYTE REGISTER

bit 7-0 HADR<15:8>: Scan End Address bits^(1, 2) Most Significant bits of the address at the end of the designated scan

Note 1: Registers SCANHADRU/H/L form a 22-bit value, but are not guarded for atomic or asynchronous access; registers should only be read or written while SGO = 0 (SCANCON0 register).

2: While SGO = 1 (SCANCON0 register), writing to this register is ignored.

REGISTER 14-17: SCANHADRL: SCAN HIGH ADDRESS LOW BYTE REGISTER

R/W-1/1	R/W-1/1	R/W-1/1	R/W-1/1	R/W-1/1	R/W-1/1	R/W-1/1	R/W-1/1				
HADR<7:0> ^(1, 2)											
bit 7							bit 0				
Legend:											
R = Readable bit		W = Writable bit	t	U = Unimpleme	ented bit, read as	ʻ0'					
u = Bit is unchang	ged	x = Bit is unkno	wn	-n/n = Value at	POR and BOR/V	alue at all other I	Resets				
'1' = Bit is set		'0' = Bit is cleare	ed								

bit 7-0 HADR<7:0>: Scan End Address bits^(1, 2)

Least Significant bits of the address at the end of the designated scan

- **Note 1:** Registers SCANHADRU/H/L form a 22-bit value, but are not guarded for atomic or asynchronous access; registers should only be read or written while SGO = 0 (SCANCON0 register).
 - 2: While SGO = 1 (SCANCON0 register), writing to this register is ignored.

15.0 DIRECT MEMORY ACCESS (DMA)

15.1 Introduction

The Direct Memory Access (DMA) module is designed to service data transfers between different memory regions directly without intervention from the CPU. By eliminating the need for CPU-intensive management of handling interrupts intended for data transfers, the CPU now can spend more time on other tasks.

PIC18(L)F25/26K83 family has two DMA modules which can be independently programmed to transfer data between different memory locations, move different data sizes, and use a wide range of hardware triggers to initiate transfers. The two DMA registers can even be programmed to work together, in order to carry out more complex data transfers without CPU overhead.

Key features of the DMA module include:

- Support access to the following memory regions:
 - GPR and SFR space (R/W)
 - Program Flash Memory (R only)
 - Data EEPROM Memory (R only)
- Programmable priority between the DMA and CPU Operations. Refer to **Section 3.1 "System Arbitration"** for details.
- Programmable Source and Destination address modes
 - Fixed address
 - Post-increment address
 - Post-decrement address
- Programmable Source and Destination sizes
- Source and destination pointer register, dynamically updated and reloadable
- Source and destination count register, dynamically updated and reloadable
- Programmable auto-stop based on Source or Destination counter
- · Software triggered transfers
- Multiple user selectable sources for hardware triggered transfers
- Multiple user selectable sources for aborting DMA transfers

15.2 DMA Registers

The operation of the DMA module has the following registers:

- Control registers (DMAxCON0, DMAxCON1)
- Data buffer register (DMAxBUF)
- Source Start Address Register (DMAxSSAU:H:L)
- Source Pointer Register (DMAxSPTRU:H:L)
- Source Message Size Register (DMAxSSZH:L)
- Source Count Register (DMAxSCNTH:L)
- Destination Start Address Register (DMAxDSAH:L)
- Destination Pointer Register (DMAxDPTRH:L)
- Destination Message Size Register (DMAxDSZH:L)
- Destination Count Register (DMAxDCNTH:L)
- Start Interrupt Request Source Register (DMAxSIRQ)
- Abort Interrupt Request Source Register (DMAxAIRQ)

These registers are detailed in Section 15.13 "Register definitions: DMA".

-				(- /	
PWM Frequency	1.22 kHz	4.88 kHz	19.53 kHz	78.12 kHz	156.3 kHz	208.3 kHz
Timer Prescale	16	4	1	1	1	1
T2PR Value	0xFF	0xFF	0xFF	0x3F	0x1F	0x17
Maximum Resolution (bits)	10	10	10	8	7	6.6

TABLE 23-2:EXAMPLE PWM FREQUENCIES AND RESOLUTIONS (Fosc = 20 MHz)

TABLE 23-3:EXAMPLE PWM FREQUENCIES AND RESOLUTIONS (Fosc = 8 MHz)

PWM Frequency	1.22 kHz	4.90 kHz	19.61 kHz	76.92 kHz	153.85 kHz	200.0 kHz
Timer Prescale	16	4	1	1	1	1
T2PR Value	0x65	0x65	0x65	0x19	0x0C	0x09
Maximum Resolution (bits)	8	8	8	6	5	5

23.4.7 OPERATION IN SLEEP MODE

In Sleep mode, the T2TMR register will not increment and the state of the module will not change. If the CCPx pin is driving a value, it will continue to drive that value. When the device wakes up, T2TMR will continue from its previous state.

23.4.8 CHANGES IN SYSTEM CLOCK FREQUENCY

The PWM frequency is derived from the system clock frequency. Any changes in the system clock frequency will result in changes to the PWM frequency. See Section 7.0 "Oscillator Module (with Fail-Safe Clock Monitor)" for additional details.

23.4.9 EFFECTS OF RESET

Any Reset will force all ports to Input mode and the CCP registers to their Reset states.

R/W-x/u	R/W-x/u	R/W-x/u	R/W-x/u	R/W-x/u	R/W-x/u	R/W-x/u	R/W-x/u
			DC·	<9:2>			
bit 7							bit 0
Legend:							
R = Readable	bit	W = Writable b	it	U = Unimplem	nented bit, read	l as '0'	
u = Bit is unch	anged	x = Bit is unkno	own	-n/n = Value a	t POR and BO	R/Value at all o	other Resets
'1' = Bit is set		'0' = Bit is clear	red				

REGISTER 24-3: PWMxDCH: PWM DUTY CYCLE HIGH BITS

bit 7-0 **DC<9:2>:** PWM Duty Cycle Most Significant bits These bits are the MSbs of the PWM duty cycle. The two LSbs are found in PWMxDCL Register.

REGISTER 24-4: PWMxDCL: PWM DUTY CYCLE LOW BITS

R/W-x/u	R/W-x/u	U-0	U-0	U-0	U-0	U-0	U-0
DC<1:0>		—	—	—	—	—	—
bit 7							bit 0

Legend:		
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'
u = Bit is unchanged	x = Bit is unknown	-n/n = Value at POR and BOR/Value at all other Resets
'1' = Bit is set	'0' = Bit is cleared	

bit 7-6DC<1:0>: PWM Duty Cycle Least Significant bits
These bits are the LSbs of the PWM duty cycle. The MSbs are found in PWMxDCH Register.bit 5-0Unimplemented: Read as '0'

TABLE 24-3: SUMMARY OF REGISTERS ASSOCIATED WITH PWM

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Register on Page
PWMxCON	EN	—	OUT	POL	—	—			344
PWMxDCH				DC<9	:2>				346
PWMxDCL	DC<1:0> — —				_	_	_	-	346
CCPTMRS1	P8TSE	SEL<1:0> P7TSEL<1:0>			P6TSE	L<1:0>	P5TSE	L<1:0>	345

Legend: - = Unimplemented locations, read as '0', u = unchanged, x = unknown. Shaded cells are not used by the PWM.

25.6.3 PERIOD AND DUTY CYCLE MODE

In Duty Cycle mode, either the duty cycle or period (depending on polarity) of the SMTx_signal can be acquired relative to the SMT clock. The CPW register is updated on a falling edge of the signal, and the CPR register is updated on a rising edge of the signal, along with the SMTxTMR resetting to 0x0001. In addition, the GO bit is reset on a rising edge when the SMT is in Single Acquisition mode. See Figure 25-6 and Figure 25-7.

REGISTER 33-9: I2CxCNT: I²C BYTE COUNT REGISTER

R/W-x/u	R/W-x/u	R/W-x/u	R/W-x/u	R/W-x/u	R/W-x/u	R/W-x/u	R/W-x/u
			CNT	<7:0>			
bit 7							bit 0
Legend:							
R = Readable b	bit	W = Writable bi	t	U = Unimpler	nented bit, read	as '0'	

u = Bit is unchanged x = Bit is unknown		-n/n = Value at POR and BOR/Value at all other Resets		
'1' = Bit is set	'0' = Bit is cleared	HS = Hardware set	HC = Hardware clear	

bit 7-0 CNT<7:0>: I²C Byte Count Register bits

If receiving data,

decremented 8th SCL edge, when a new data byte is loaded into I2CxRXB

If transmitting data,

decremented 9th SCL edge, when a new data byte is moved from I2CxTXB

CNTIF flag is set on 9th falling SCL edge, when I2CxCNT = 0. (Byte count cannot decrement past '0')

Note 1: It is recommended to write this register only when the module is IDLE (MMA = 0, SMA = 0) or when clock stretching (CSTR = 1 || MDR = 1).

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	
B5IE ⁽²⁾	B4IE ⁽²⁾	B3IE ⁽²⁾	B2IE ⁽²⁾	B1IE ⁽²⁾	B0IE ⁽²⁾	RXB1IE ⁽²⁾	RXB0IE ⁽²⁾	
bit 7							bit 0	
Legend:								
R = Readable	bit	W = Writable	bit	U = Unimplemented bit, read as '0'				
-n = Value at P	POR	'1' = Bit is set		'0' = Bit is cleared		x = Bit is unknown		
bit 7-2 B<5:0>IE: Programmable Transmit/Receive 1 = Interrupt is enabled 0 = Interrupt is disabled bit 1-0 RXB<1:0>IE: Dedicated Receive Buffer 1-1-1 1 = Interrupt is enabled 0 = Interrupt is disabled					bits ⁽²⁾			

REGISTER 34-60: BIE0: BUFFER INTERRUPT ENABLE REGISTER 0⁽¹⁾

Note 1: This register is available in Mode 1 and 2 only.

2: Either TXBnIE or RXBnIE, in the PIE5 register, must be set to get an interrupt.

REGISTER 37-2: ADCON1: ADC CONTROL REGISTER 1

R/W-0/0	R/W-0/0	R/W-0/0	U-0	U-0	U-0	U-0	R/W-0/0
PPOL	IPEN	GPOL	—	-	-	-	DSEN
bit 7							bit 0

Legend:		
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'
u = Bit is unchanged	x = Bit is unknown	-n/n = Value at POR and BOR/Value at all other Resets
'1' = Bit is set	'0' = Bit is cleared	

bit 7 **PPOL:** Precharge Polarity bit If PRE>0x00:

PPOL	Action During 1st Precharge Stage				
FFUL	External (selected analog I/O pin)	Internal (AD sampling capacitor)			
1	Connected to VDD	C _{HOLD} connected to Vss			
0	Connected to Vss	C _{HOLD} connected to VDD			

- Otherwise:
- The bit is ignored

bit 6 IPEN: A/D Inverted Precharge Enable bit

If DSEN = 1

- 1 = The precharge and guard signals in the second conversion cycle are the opposite polarity of the first cycle
- 0 = Both Conversion cycles use the precharge and guards specified by ADPPOL and ADGPOL

Otherwise:

The bit is ignored

bit 5 **GPOL:** Guard Ring Polarity Selection bit

- 1 = ADC guard Ring outputs start as digital high during Precharge stage
- 0 = ADC guard Ring outputs start as digital low during Precharge stage

bit 4-1 Unimplemented: Read as '0'

bit 0 DSEN: Double-sample enable bit

- 1 = Two conversions are performed on each trigger. Data from the first conversion appears in PREV
- 0 = One conversion is performed for each trigger

REGISTER 37-24: ADACCU: ADC ACCUMULATOR REGISTER UPPER

U-0	U-0	U-0	U-0	U-0	U-0	R/W-x/x	R/W-x/x
	_	—	—	—	—	ACC<	17:16>
bit 7							bit 0
Legend:							
R = Readable bit W = Writable		bit	U = Unimpler	nented bit, read	1 as '0'		
u = Bit is unchanged		x = Bit is unknown		-n/n = Value at POR and BOR/Value at all other Resets			
'1' = Bit is set		'0' = Bit is clea	ared				

bit 7-2 Unimplemented: Read as '0'

Γ.

bit 1-0 ACC<17:16>: ADC Accumulator MSB. Upper two bits of accumulator value. See Table 37-2 for more details.

REGISTER 37-25: ADACCH: ADC ACCUMULATOR REGISTER HIGH

R/W-x/x	R/W-x/x	R/W-x/x	R/W-x/x	R/W-x/x	R/W-x/x	R/W-x/x	R/W-x/x	
	ACC<15:8>							
bit 7							bit 0	

Legend:		
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'
u = Bit is unchanged	x = Bit is unknown	-n/n = Value at POR and BOR/Value at all other Resets
'1' = Bit is set	'0' = Bit is cleared	

bit 7-0 ACC<15:8>: ADC Accumulator middle bits. Middle eight bits of accumulator value. See Table 37-2 for more details.

REGISTER 37-26: ADACCL: ADC ACCUMULATOR REGISTER LOW

R/W-x/x	R/W-x/x	R/W-x/x	R/W-x/x	R/W-x/x	R/W-x/x	R/W-x/x	R/W-x/x	
	ACC<7:0>							
bit 7							bit 0	
Legend:								

Legend:		
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'
u = Bit is unchanged	x = Bit is unknown	-n/n = Value at POR and BOR/Value at all other Resets
'1' = Bit is set	'0' = Bit is cleared	

bit 7-0 ACC<7:0>: ADC Accumulator LSB. Lower eight bits of accumulator value. See Table 37-2 for more details.

© 2017 Microchip Technology Inc.

Branch if Not Zero

 $(PC) + 2 + 2n \rightarrow PC$

0001

incremented to fetch the next instruction, the new address will be

If the ZERO bit is '0', then the program

The 2's complement number '2n' is added to the PC. Since the PC will have

nnnn

nnnn

BNZ n

None 1110

will branch.

 $-128 \le n \le 127$ if ZERO bit is '0'

BNOV	Branch if	Not Overflo	w	BNZ
Syntax:	BNOV n			Syntax:
Operands:	-128 ≤ n ≤ 1	27		Operands:
Operation:	if OVERFLO (PC) + 2 + 2			Operation:
Status Affected:	None			Status Affected:
Encoding:	1110	0101 nni	nn nnnn	Encoding:
Description:	program wil The 2's con added to the incremented instruction,	nplement num e PC. Since th d to fetch the r the new addre n. This instruct	Description:	
Words:	1			Words:
Cycles:	1(2)			Cycles:
Q Cycle Activity: If Jump:				Q Cycle Activity: If Jump:
Q1	Q2	Q3	Q4	Q1
Decode	Read literal 'n'	Process Data	Write to PC	Decode F
No operation	No operation	No operation	No operation	No operation
If No Jump:				If No Jump:
Q1	Q2	Q3	Q4	Q1
Decode	Read literal 'n'	Process Data	No operation	Decode F
Example: Before Instruc PC After Instructio If OVERF PC If OVERF PC	= ado n FLOW = 0; = ado FLOW = 1;	BNOV Jump dress (HERE dress (Jump dress (HERE)	Example: Before Instructio PC After Instruction If ZERO PC If ZERO PC

PC + 2 + 2n. This instruction is then a 2-cycle instruction. 1 1(2) Q3 Q2 Q4 Read literal Process Write to PC 'n' Data No No No operation operation operation Q2 Q3 Q4 Read literal Process No 'n' Data operation HERE BNZ Jump on address (HERE) = = 0;

address (Jump)

1; address (HERE + 2)

=

=

=

DCFSNZ	Decremer	Decrement f, skip if not 0				
Syntax:	DCFSNZ	f {,d {,a}}				
Operands:	$0 \le f \le 255$ $d \in [0,1]$ $a \in [0,1]$					
Operation:	()	(f) – 1 → dest, skip if result \neq 0				
Status Affected:	None					
Encoding:	0100	11da fff	f ffff			
Description:	Description: The contents of register 'f' are decremented. 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 2-cycle instruction. If 'a' is '0', the Access Bank is selected. If 'a' is '1', the BSR is used to select the GPR bank. If 'a' is '0' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever f ≤ 95 (5Fh). See Sec- tion 42.2.3 "Byte-Oriented and Bit- Oriented Instructions in Indexed Lit- eral Offset Mode" for details.					
Words:	1		ano.			
Cycles:		cycles if skip an a 2-word instr				
Q Cycle Activity:						
Q1	Q2	Q3	Q4			
Decode	Read	Process	Write to			
lf skip:	register 'f'	Data	destination			
Q1	Q2	Q3	Q4			
No	No	No	No			
operation	operation	operation	operation			
If skip and followe			04			
Q1 No	Q2 No	Q3 No	Q4 No			
operation	operation	operation	operation			
No	No	No	No			
operation	operation	operation	operation			
Example:	ZERO S	:	P, 1, 0			
Before Instruc TEMP	tion =	?				
After Instruction		•				
	=	TEMP – 1,				
If TEMP PC	=	0; Address (2	LERO)			
If TEMP PC	≠ =	0; Address (1				

GOTO		Unconditional Branch						
Syntax:		GOTO k	GOTO k					
Operands:		$0 \le k \le 10^4$	48575					
Operation:		$k \rightarrow PC < 20:1 >$						
Status Affected:		None						
Encoding: 1st word (k<7:0>) 2nd word(k<19:8>)		1110 1111	1111 k ₁₉ kkk	k ₇ kkk kkkk	kkkk ₀ kkkk ₈			
Desc	ription:	GOTO allows an unconditional branch anywhere within entire 2-Mbyte memory range. The 20-bit value 'k' is loaded into PC<20:1>. GOTO is always a 2-cycle instruction.						
Words:		2						
Cycles:		2						
Q Cycle Activity:								
Q1		Q2 Q3			Q4			
	Decode	Read literal 'k'<7:0>,	No opera	tion	Read literal k'<19:8>, Vrite to PC			
	No	No	No		No			
	operation	operation	opera	tion	operation			
<u>Exan</u>	nple: After Instructio PC =	GOTO THE n Address (1						

SUBFWB	Subtract	f from W wi	ith borrow	SUBLW
Syntax:	SUBFWB	f {,d {,a}}		Syntax:
Operands:	$0 \le f \le 255$			Operands:
	d ∈ [0,1] a ∈ [0,1]			Operation:
Operation:		$(\overline{C}) \rightarrow \text{dest}$		Status Affected:
Status Affected:	N, OV, C,	. ,		Encoding:
Encoding:	0101	01da ffi	ff ffff	Description
Description:		egister 'f' and (
Description.		om W (2's con	-	Words:
	method). It	f'd' is '0', the re	esult is stored	Cycles:
	in W. If 'd' register 'f'	is '1', the resu	llt is stored in	Q Cycle Activity
		the Access Ba	ank is	Q1
		f 'a' is '1', the l	BSR is used	Decode
		ne GPR bank. and the extend	ed instruction	
	set is enab	oled, this instru	uction	Example 1:
	•	n Indexed Liter		Before Insti W
		g mode whene). See Sectio	С	
	42.2.3 "By	te-Oriented a	nd Bit-Ori-	After Instru W
		ructions in Ind de" for details.		C Z
Words:	1			Ň
	1		Example 2:	
Cycles:	I			Before Inst
Q Cycle Activity: Q1	Q2	Q3	Q4	W C
Decode	Read	Process	Write to	After Instru W
	register 'f'	Data	destination	С
Example 1:	SUBFWB	REG, 1, 0		Z N
Before Instruct				Example 3:
REG W	= 3 = 2			Before Inst
C After Instructio	= 1			W C
REG	= FF			After Instru
W C	= 2 = 0			W C
Z N	= 0 = 1 ; re	sult is negative	2	Ž N
Example 2:	SUBFWB	REG, 0, 0		IN IN
Before Instruct				
REG W	= 2 = 5			
C After Instructio	= 1			
REG	= 2			
W C	= 3 = 1			
Z N	= 0 = 0 : re	sult is positive		
Example 3:	SUBFWB	REG, 1, 0		
Before Instruct	tion			
REG W	= 1 = 2			
С	= 0			
After Instructio REG	n = 0			
W C	= 2 = 1			
Z	= 1 ; re	sult is zero		
N	= 0			

SUBLW			Subtract W from literal						
Syntax:			SUBLW k						
Operands:			$0 \le k \le 255$						
Oper	ation:		$k - (W) \rightarrow$	W					
			N, OV, C, DC, Z						
Enco	ding:		0000 1000 kkkk kkkk						
Operands: Operation: Status Affected: Encoding: Description Words: Cycles: Q Cycle Activity: Q1 Decode I Example 1: Before Instruction W = C = After Instruction		W is subtracted from the 8-bit literal 'k'. The result is placed in W.							
Status Affected: Encoding: Description Words: Cycles: Q Cycle Activity: Q1 Decode Example 1: Before Instructior W = C =			1						
Cycle	es:		1						
QC	ycle Activity:								
	Q1		Q2	Q3		Q4			
	Decode		Read teral 'k'	Proce Data		W	rite to W		
Exan	nple 1:		SUBLW ()2h					
	W C After Instructio W C	= = = = =	01h ? 01h 1 ; re 0	esult is po	ositive				
Exan	C = ? After Instruction $W = 01h$ $C = 1 ; result is positive$ $Z = 0$ $N = 0$ Example 2: SUBLW 02h Before Instruction $W = 02h$ $C = ?$ After Instruction $W = 00h$								
W = 02h C = ? After Instruction									
Exan	nple <u>3</u> :		SUBLW ()2h					
	Before Instruc W C After Instructic W C Z N	= =		2's comp esult is n					

Addr	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Register on page
3F96h	T6CLK	—	—	_	—		(CS		300
3F95h	T6HLT	PSYNC	CKPOL	CKSYNC		MODE				
3F94h	T6CON	ON		CKPS			OU	ITPS		323
3F93h	T6PR				PR	6				322
3F92h	T6TMR				TMF	76				322
3F91h	ECANCON	MDSEL1	MDSEL0	FIFOWM	EWIN4	EWIN3	EWIN2	EWIN1	EWIN0	607
3F90h	COMSTAT	RXB00VFL	RXB10VFL	TXBO	TXBP	RXBP	TXWARN	RXWARN	EWARN	608
3F90h	COMSTAT	_	RXBnOVFL	TXBO	TXBP	RXBP	TXWARN	RXWARN	EWARN	608
3F90h	COMSTAT	FIFOEMPTY	RXBnOVFL	ТХВО	TXBP	RXBP	TXWARN	RXWARN	EWARN	608
3F8Fh	CANCON	REQOP2	REQOP1	REQOP0	ABAT	WIN2	WIN1	WIN0	_	603
3F8Fh	CANCON	REQOP2	REQOP1	REQOP0	ABAT	_	_	_	_	603
3F8Fh	CANCON	REQOP2	REQOP1	REQOP0	ABAT	FP3	FP2	FP1	FP0	603
3F8Eh	CANSTAT	OPMODE2	OPMODE1	OPMODE0	_	ICODE2	ICODE1	ICODE0		604
3F8Eh	CANSTAT	OPMODE2	OPMODE1	OPMODE0	EICODE4	EICODE3	EICODE2	EICODE1	EICODE0	604
3F8Dh	RXB0D7	RXB0Dm7	RXB0Dm6	RXB0Dm5	RXB0Dm4	RXB0Dm3	RXB0Dm2	RXB0Dm1	RXB0Dm0	620
3F8Ch	RXB0D6	RXB0Dm7	RXB0Dm6	RXB0Dm5	RXB0Dm4	RXB0Dm3	RXB0Dm2	RXB0Dm1	RXB0Dm0	620
3F8Bh	RXB0D5	RXB0Dm7	RXB0Dm6	RXB0Dm5	RXB0Dm4	RXB0Dm3	RXB0Dm2	RXB0Dm1	RXB0Dm0	620
3F8Ah	RXB0D4	RXB0Dm7	RXB0Dm6	RXB0Dm5	RXB0Dm4	RXB0Dm3	RXB0Dm2	RXB0Dm1	RXB0Dm0	620
3F89h	RXB0D3	RXB0Dm7	RXB0Dm6	RXB0Dm5	RXB0Dm4	RXB0Dm3	RXB0Dm2	RXB0Dm1	RXB0Dm0	620
3F88h	RXB0D2	RXB0Dm7	RXB0Dm6	RXB0Dm5	RXB0Dm4	RXB0Dm3	RXB0Dm2	RXB0Dm1	RXB0Dm0	620
3F87h	RXB0D1	RXB0Dm7	RXB0Dm6	RXB0Dm5	RXB0Dm4	RXB0Dm3	RXB0Dm2	RXB0Dm1	RXB0Dm0	620
3F86h	RXB0D0	RXB0Dm7	RXB0Dm6	RXB0Dm5	RXB0Dm4	RXB0Dm3	RXB0Dm2	RXB0Dm1	RXB0Dm0	620
3F85h	RXB0DLC	_	RXRTR	RB1	RB0	DLC3	DLC2	DLC1	DLC0	620
3F84h	RXB0EIDL	EID7	EID6	EID5	EID4	EID3	EID2	EID1	EID0	620
3F83h	RXB0EIDH	EID15	EID14	EID13	EID12	EID11	EID10	EID9	EID8	620
3F82h	RXB0SIDL	SID2	SID1	SID0	SRR	EXID		EID17	EID16	620
3F81h	RXB0SIDH	SID10	SID9	SID8	SID7	SID6	SID5	SID4	SID3	620
3F80h	RXB0CON	RXFUL	RXM1	RXM0		RXRTRRO	RXB0DBEN	JTOFF	FILHIT0	620
3F80h	RXB0CON	RXFUL	RXM1	RTRRO	FILHITF4	FILHIT3	FILHIT2	FILHIT1	FILHITO	620
3F7Fh	CCP1CAP	_		_	_			<3:0>		338
3F7Eh	CCP1CON	EN		OUT	FMT			ODE		335
3F7Dh	CCPR1H	2			RH	1		002		339
3F7Ch	CCPR1L				RL					338
3F7Bh	CCP2CAP	CTS<3:0>					338			
3F7Ah	CCP2CON	EN		OUT	FMT	MODE				
3F79h	CCPR2H	RH					335 339			
3F78h	CCPR2L				RL					338
3F77h	CCP3CAP	CTS<3:0>					338			
3F76h	CCP3CON	EN	_	OUT	FMT					335
3F75h	CCPR3H	EN – OUT FMT MODE					339			
3F74h	CCPR3L	RL					338			
3F73h	CCP4CAP	_	_	_	_	CTS<3:0>				338
3F72h	CCP4CON	EN		OUT	FMT			DDE		335
3F71h	CCPR4H			001	RH	I	IVIC			339
3F70h	CCPR4L				RI					338
3F6Fh									556	
3F6Eh	PWM5CON	EN	Unimplemented EN — OUT POL — — — —				344			
3F6Dh	PWM5CON PWM5DCH	DC9	DC8	DC7	DC6	DC5	DC4	DC3	DC2	344
					DC0	000	004	003	002	
3F6Ch	PWM5DCL	DC1	DC0							346

TABLE 43-1: REGISTER FILE SUMMARY FOR PIC18(L)F25/26K83 DEVICES (CONTINUED)

Note 1: Not present in LF devices.

Note the following details of the code protection feature on Microchip devices:

- · Microchip products meet the specification contained in their particular Microchip Data Sheet.
- Microchip believes that its family of products is one of the most secure families of its kind on the market today, when used in the intended manner and under normal conditions.
- There are dishonest and possibly illegal methods used to breach the code protection feature. All of these methods, to our knowledge, require using the Microchip products in a manner outside the operating specifications contained in Microchip's Data Sheets. Most likely, the person doing so is engaged in theft of intellectual property.
- Microchip is willing to work with the customer who is concerned about the integrity of their code.
- Neither Microchip nor any other semiconductor manufacturer can guarantee the security of their code. Code protection does not mean that we are guaranteeing the product as "unbreakable."

Code protection is constantly evolving. We at Microchip are committed to continuously improving the code protection features of our products. Attempts to break Microchip's code protection feature may be a violation of the Digital Millennium Copyright Act. If such acts allow unauthorized access to your software or other copyrighted work, you may have a right to sue for relief under that Act.

Information contained in this publication regarding device applications and the like is provided only for your convenience and may be superseded by updates. It is your responsibility to ensure that your application meets with your specifications. MICROCHIP MAKES NO REPRESENTATIONS OR WARRANTIES OF ANY KIND WHETHER EXPRESS OR IMPLIED, WRITTEN OR ORAL, STATUTORY OR OTHERWISE, RELATED TO THE INFORMATION, INCLUDING BUT NOT LIMITED TO ITS CONDITION, QUALITY, PERFORMANCE, MERCHANTABILITY OR FITNESS FOR PURPOSE. Microchip disclaims all liability arising from this information and its use. Use of Microchip devices in life support and/or safety applications is entirely at the buyer's risk, and the buyer agrees to defend, indemnify and hold harmless Microchip from any and all damages, claims, suits, or expenses resulting from such use. No licenses are conveyed, implicitly or otherwise, under any Microchip intellectual property rights unless otherwise stated.

Microchip received ISO/TS-16949:2009 certification for its worldwide headquarters, design and wafer fabrication facilities in Chandler and Tempe, Arizona; Gresham, Oregon and design centers in California and India. The Company's quality system processes and procedures are for its PIC® MCUs and dsPIC® DSCs, KEEL0Q® code hopping devices, Serial EEPROMs, microperipherals, nonvolatile memory and analog products. In addition, Microchip's quality system for the design and manufacture of development systems is ISO 9001:2000 certified.

QUALITY MANAGEMENT SYSTEM CERTIFIED BY DNV = ISO/TS 16949=

Trademarks

The Microchip name and logo, the Microchip logo, AnyRate, AVR, AVR logo, AVR Freaks, BeaconThings, BitCloud, chipKIT, chipKIT logo, CryptoMemory, CryptoRF, dsPIC, FlashFlex, flexPWR, Heldo, JukeBlox, KEELOQ, KEELOQ logo, Kleer, LANCheck, LINK MD, maXStylus, maXTouch, MediaLB, megaAVR, MOST, MOST logo, MPLAB, OptoLyzer, PIC, picoPower, PICSTART, PIC32 logo, Prochip Designer, QTouch, RightTouch, SAM-BA, SpyNIC, SST, SST Logo, SuperFlash, tinyAVR, UNI/O, and XMEGA are registered trademarks of Microchip Technology Incorporated in the U.S.A. and other countries.

ClockWorks, The Embedded Control Solutions Company, EtherSynch, Hyper Speed Control, HyperLight Load, IntelliMOS, mTouch, Precision Edge, and Quiet-Wire are registered trademarks of Microchip Technology Incorporated in the U.S.A.

Adjacent Key Suppression, AKS, Analog-for-the-Digital Age, Any Capacitor, AnyIn, AnyOut, BodyCom, CodeGuard, CryptoAuthentication, CryptoCompanion, CryptoController, dsPICDEM, dsPICDEM.net, Dynamic Average Matching, DAM, ECAN, EtherGREEN, In-Circuit Serial Programming, ICSP, Inter-Chip Connectivity, JitterBlocker, KleerNet, KleerNet logo, Mindi, MiWi, motorBench, MPASM, MPF, MPLAB Certified logo, MPLIB, MPLINK, MultiTRAK, NetDetach, Omniscient Code Generation, PICDEM, PICDEM.net, PICkit, PICtail, PureSilicon, QMatrix, RightTouch logo, REAL ICE, Ripple Blocker, SAM-ICE, Serial Quad I/O, SMART-I.S., SQI, SuperSwitcher, SuperSwitcher II, Total Endurance, TSHARC, USBCheck, VariSense, ViewSpan, WiperLock, Wireless DNA, and ZENA are trademarks of Microchip Technology Incorporated in the U.S.A. and other countries.

 $\ensuremath{\mathsf{SQTP}}$ is a service mark of Microchip Technology Incorporated in the U.S.A.

Silicon Storage Technology is a registered trademark of Microchip Technology Inc. in other countries.

GestIC is a registered trademark of Microchip Technology Germany II GmbH & Co. KG, a subsidiary of Microchip Technology Inc., in other countries.

All other trademarks mentioned herein are property of their respective companies.

© 2017, Microchip Technology Incorporated, All Rights Reserved. ISBN: 978-1-5224-2047-7