



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	Obsolete
Core Processor	ACE1502
Core Size	8-Bit
Speed	25MHz
Connectivity	-
Peripherals	Brown-out Detect/Reset, LVD, POR, PWM, WDT
Number of I/O	6
Program Memory Size	2KB (2K x 8)
Program Memory Type	EEPROM
EEPROM Size	64 x 8
RAM Size	64 x 8
Voltage - Supply (Vcc/Vdd)	1.8V ~ 3.6V
Data Converters	-
Oscillator Type	Internal
Operating Temperature	-40°C ~ 125°C (TA)
Mounting Type	Through Hole
Package / Case	8-DIP (0.400", 10.16mm)
Supplier Device Package	8-DIP
Purchase URL	https://www.e-xfl.com/product-detail/onsemi/ace1502vn

Figure 2. ACEx Application Example (Remote Keyless Entry)

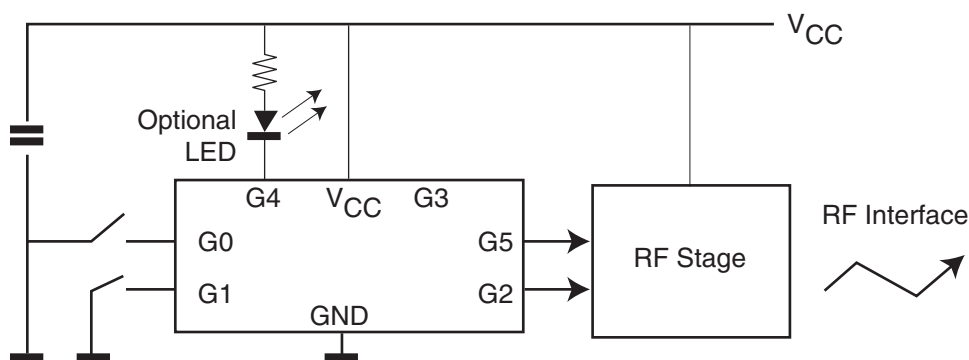
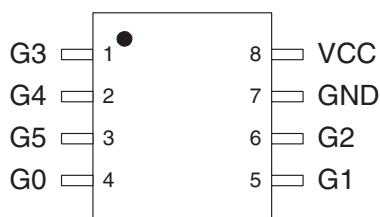


Figure 3. ACE1502 8-pin SOIC and DIP Device Pinout

a) Normal Mode Operation



b) Programming Mode Operation

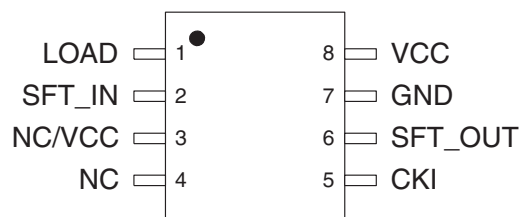
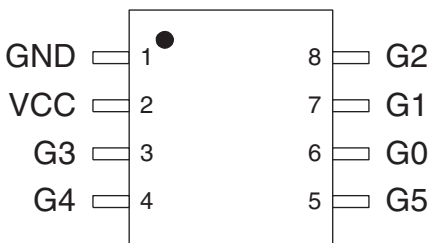


Figure 4. ACE1502 8-pin TSSOP Device Pinout

a) Normal Mode Operation



b) Programming Mode Operation

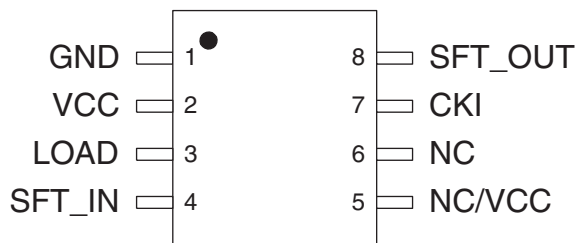
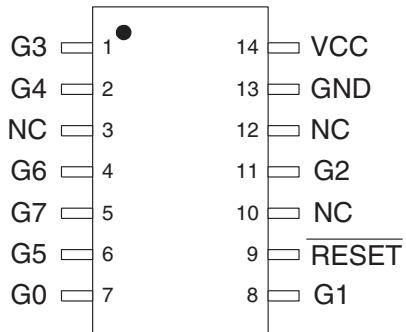


Figure 5. ACE1502 14-pin SOIC, TSSOP and DIP Device Pinout

a) Normal Mode Operation



b) Programming Mode Operation

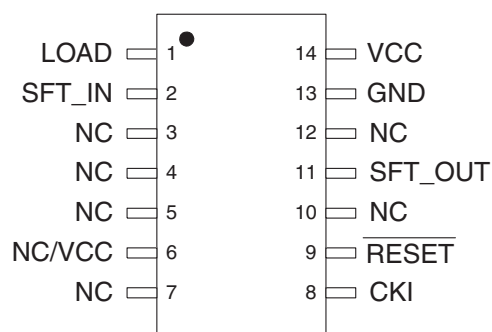


Figure 8. Icc Active

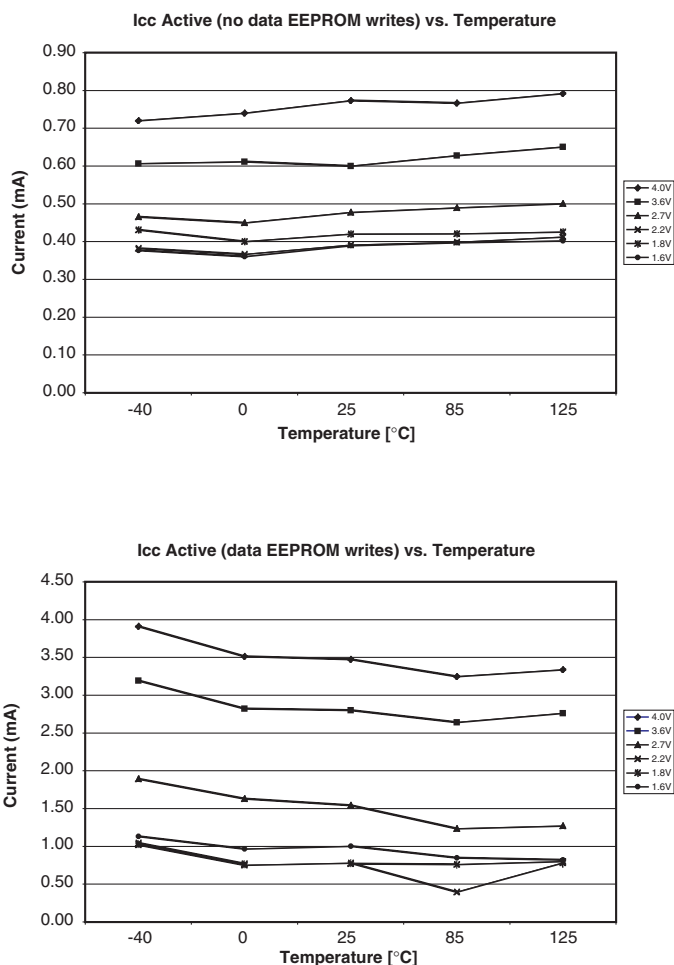
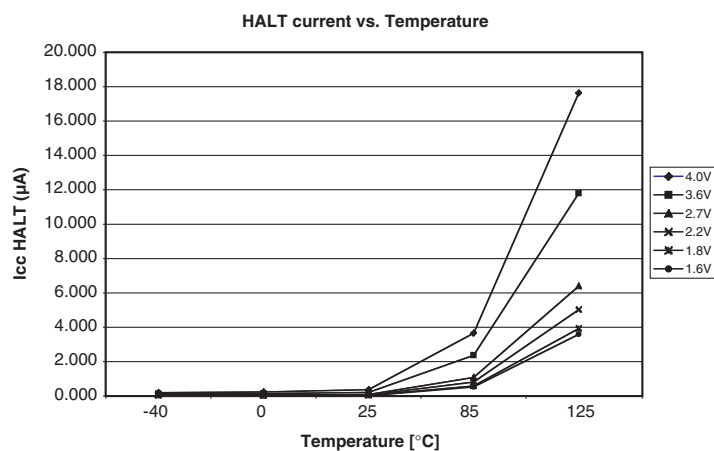


Figure 9. HALT Mode Currents



3. Arithmetic Controller Core

The ACE[™] microcontroller core is specifically designed for low cost applications involving bit manipulation, shifting and block encryption. It is based on a modified Harvard architecture meaning peripheral, I/O, and RAM locations are addressed separately from instruction data.

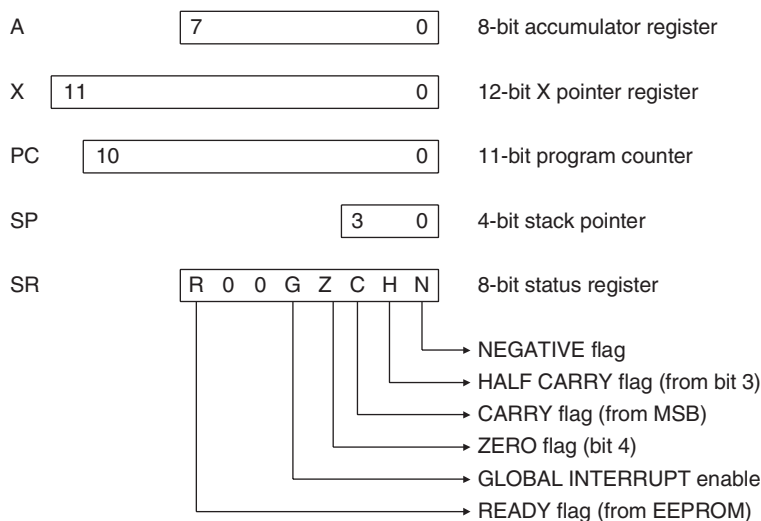
The core differs from the traditional Harvard architecture by aligning the data and instruction memory sequentially. This allows the X-pointer (12-bits) to point to any memory location in either segment of the memory map. This modification improves

the overall code efficiency of the ACE[™] microcontroller and takes advantage of the flexibility found on Von Neumann style machines.

3.1 CPU Registers

The ACE[™] microcontroller has five general-purpose registers. These registers are the Accumulator (A), X-Pointer (X), Program Counter (PC), Stack Pointer (SP), and Status Register (SR). The X, SP, and SR registers are all memory-mapped.

Figure 12. Programming Model



3.1.1 Accumulator (A)

The Accumulator is a general-purpose 8-bit register that is used to hold data and results of arithmetic calculations or data manipulations.

3.1.2 X-Pointer (X)

The X-Pointer register allows for a 12-bit indexing value to be added to an 8-bit offset creating an effective address used for reading and writing between the entire memory space. (Software can only read from code EEPROM.) This provides software with the flexibility of storing lookup tables in the code EEPROM memory space for the core's accessibility during normal operation.

The ACE[™] core allows software to access the entire 12-bit X-Pointer register using the special X-pointer instructions e.g. LD X, #000H. (See Table 8.) However, software may also access the register through any of the memory-mapped instructions using the XHI (X[11:8]) and XLO (X[7:0]) variables located at 0xBE and 0xBF, respectively. (See Table 10.)

The X register is divided into two sections. The 11 least significant bits (LSBs) of the register is the address of the program or data memory space. The most significant bit (MSB) of the register is write only and selects between the data (0x000 to 0x0FF) or program (0x800 to 0xFFFF) memory space.

Example: If Bit 11 = 0, then the LD A, [00,X] instruction will take a value from address range 0x000 to 0x0FF and load it into A. If

Bit 11 = 1, then the LD A, [00,X] instruction will take a value from address range 0x800 to 0xFFFF and load it into A.

The X register can also serve as a counter or temporary storage register. However, this is true only for the 11-LSBs since the 12th bit is dedicated for memory space selection.

3.1.3 Program Counter (PC)

The 11-bit program counter register contains the address of the next instruction to be executed. After a reset, if in normal mode the program counter is initialized to 0x800.

3.1.4 Stack Pointer (SP)

The ACE[™] microcontroller has an automatic program stack with a 4-bit stack pointer. The stack can be initialized to any location between addresses 0x30-0x3F. Normally, the stack pointer is initialized by one of the first instructions in an application program. After a reset, the stack pointer is defaulted to 0xF pointing to address 0x3F.

The stack is configured as a data structure which decrements from high to low memory. Each time a new address is pushed onto the stack, the core decrements the stack pointer by two. Each time an address is pulled from the stack, the core increments the stack pointer by two. At any given time, the stack pointer points to the next free location in the stack.

When a subroutine is called by a jump to subroutine (JSR) instruction, the address of the instruction is automatically pushed onto the stack least significant byte first. When the

subroutine is finished, a return from subroutine (RET) instruction is executed. The RET instruction pulls the previously stacked return address from the stack and loads it into the program counter. Execution then continues at the recovered return address.

3.1.5 Status Register (SR)

The 8-bit Status register (SR) contains four condition code indicators (C, H, Z, and N), one interrupt masking bit (G), and an EEPROM write flag (R.) The condition codes are automatically updated by most instructions. (See Table 9.)

Carry/Borrow (C)

The carry flag is set if the arithmetic logic unit (ALU) performs a carry or borrow during an arithmetic operation and by its dedicated instructions. The rotate instruction operates with and through the carry bit to facilitate multiple-word shift operations. The LDC and INVC instructions facilitate direct bit manipulation using the carry flag.

Half Carry (H)

The half carry flag indicates whether an overflow has taken place on the boundary between the two nibbles in the accumulator. It is primarily used for Binary Coded Decimal (BCD) arithmetic calculation.

Zero (Z)

The zero flag is set if the result of an arithmetic, logic, or data manipulation operation is zero. Otherwise, it is cleared.

Negative (N)

The negative flag is set if the MSB of the result from an arithmetic, logic, or data manipulation operation is set to one. Otherwise, the flag is cleared. A result is said to be negative if its MSB is a one.

Interrupt Mask (G)

The interrupt request mask (G) is a global mask that disables all maskable interrupt sources. If the G Bit is cleared, interrupts can become pending, but the operation of the core continues uninterrupted. However, if the G Bit is set an interrupt is recognized. After any reset, the G bit is cleared by default and can only be set by a software instruction. When an interrupt is recognized, the G bit is cleared after the PC is stacked and the interrupt vector is fetched. Once the interrupt is serviced, a

return from interrupt instruction is normally executed to restore the PC to the value that was present before the interrupt occurred. The G bit is the reset to one after a return from interrupt is executed. Although the G bit can be set within an interrupt service routine, “nesting” interrupts in this way should only be done when there is a clear understanding of latency and of the arbitration mechanism.

3.2 Interrupt handling

When an interrupt is recognized, the current instruction completes its execution. The return address (the current value in the program counter) is pushed onto the stack and execution continues at the address specified by the unique interrupt vector (see Table 10.). This process takes five instruction cycles. At the end of the interrupt service routine, a return from interrupt (RETI) instruction is executed. The RETI instruction causes the saved address to be pulled off the stack in reverse order. The G bit is set and instruction execution resumes at the return address.

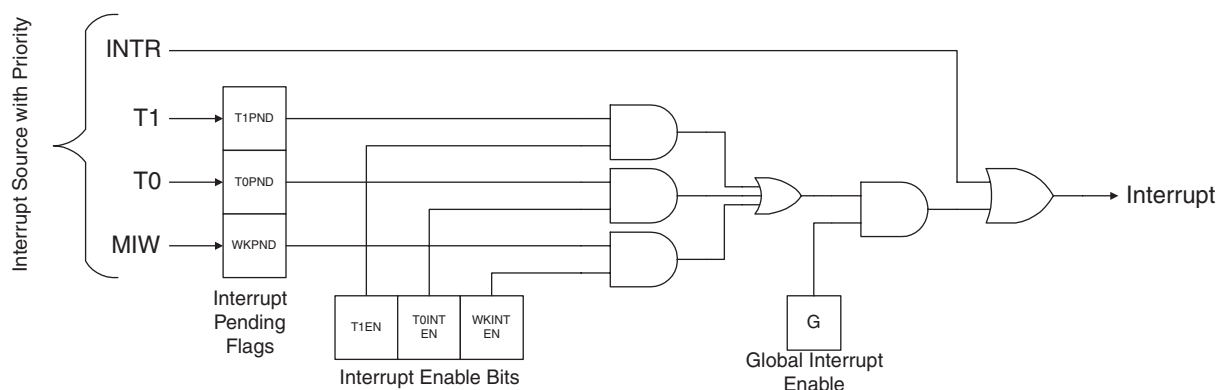
The ACEx microcontroller is capable of supporting four interrupts. Three are maskable through the G bit of the SR and the fourth (software interrupt) is not inhibited by the G bit (Figure 13.) The software interrupt is generated by the execution of the INTR instruction. Once the INTR instruction is executed, the ACEx core will interrupt whether the G bit is set or not. The INTR interrupt is executed in the same manner as the other maskable interrupts where the program counter register is stacked and the G bit is cleared. This means, if the G bit was enabled prior to the software interrupt the RETI instruction must be used to return from interrupt in order to restore the G bit to its previous state. However, if the G bit was not enabled prior to the software interrupt the RET instruction must be used.

In case of multiple interrupts occurring at the same time, the ACEx microcontroller core has prioritized the interrupts. The interrupt priority sequence is shown in Table 7.

Table 7: Interrupt Priority Sequence

Priority (4 highest, 1 lowest)	Interrupt
4	MIW (EDGEI)
3	Timer0 (TMRI0)
2	Timer1 (TMRI1)
1	Software (INTR)

Figure 13. Basic Interrupt Structure



3.3 Addressing Modes

The ACEx microcontroller has seven addressing modes indexed, indirect, direct, immediate, absolute jump, and relative jump.

Indexed

The instruction allows an 8-bit unsigned offset value to be added to the 11-LSBs of the X-pointer yielding a new effective address. This mode can be used to address either data or program memory space.

Indirect

The instruction allows the X-pointer to address any location within the data memory space.

Direct

The instruction contains an 8-bit address field that directly points to the data memory space as an operand.

Immediate

The instruction contains an 8-bit immediate field as an operand.

Inherent

This instruction has no operands associated with it.

Absolute

The instruction contains an 11-bit address that directly points to a location in the program memory space. There are two operands associated with this addressing mode. Each operand contains a byte of an address. This mode is used only for the long jump (JMP) and JSR instructions.

Relative

This mode is used for the short jump (JP) instructions where the operand is a value relative to the current PC address. With this instruction, software is limited to the number of bytes it can jump, -31 or +32.

Table 8. Instruction Addressing Modes

Instruction	Immediate			Direct	Indexed	Indirect	Inherent	Relative	Absolute
ADC	A, #			A, M	A, [# , X]	A, [X]			
ADD	A, #			A, M	A, [# , X]	A, [X]			
AND	A, #			A, M	A, [# , X]	A, [X]			
OR	A, #			A, M	A, [# , X]	A, [X]			
SUBC	A, #			A, M	A, [# , X]	A, [X]			
XOR	A, #			A, M	A, [# , X]	A, [X]			
CLR				M			A	X	
INC				M			A	X	
DEC				M			A	X	
IFEQ	A, #	X, #	M, #	A, M	A, [# , X]	A, [X]			
IFGT	A, #	X, #		A, M	A, [# , X]	A, [X]			
IFNE	A, #	X, #	M, #	A, M	A, [# , X]	A, [X]			
IFLT		X, #							
SC							no-op		
RC							no-op		
IFC							no-op		
IFNC							no-op		
INVC							no-op		
LDC				#, M					
STC				#, M					
RLC				M			A		
RRC				M			A		
LD	A, #	M, #	X, #	A, M	A, [# , X]	A, [X]			
ST				A, M	A, [# , X]	A, [X]			
NOP							no-op		
IFBIT	#, A			#, M		[#, X]			
IFNBIT	#, A			#, M		[#, X]			
SBIT				#, M		[#, X]			
RBIT				#, M		[#, X]			
JP					[#, X]			Rel	M
JSR					[#, X]				M
JMP									
RET							no-op		
RETI							no-op		
INTR							no-op		

Table 9. Instruction Cycles and Bytes

Mnemonic	Operand	Bytes	Cycles	Flags affected
ADC	A, [X]	1	1	C,H,Z,N
ADC	A, [#X]	2	3	C,H,Z,N
ADC	A, M	2	2	C,H,Z,N
ADC	A, #	2	2	C,H,Z,N
ADD	A, [X]	1	1	Z,N
ADD	A, [#X]	2	3	Z,N
ADD	A, M	2	2	Z,N
ADD	A, #	2	2	Z,N
AND	A, [X]	1	1	Z,N
AND	A, [#X]	2	3	Z,N
AND	A, M	2	2	Z,N
AND	A, #	2	2	Z,N
CLR	X	1	1	Z
CLR	A	1	1	C,H,Z,N
CLR	M	2	1	C,H,Z,N
DEC	X	1	1	Z
DEC	A	1	1	Z,N
DEC	M	2	2	Z,N
IFBIT	#, A	1	1	None
IFBIT	#, M	2	2	None
IFBIT	#, [X]	1	1	None
IFC		1	1	None
IFEQ	A, [#X]	2	3	None
IFEQ	A, [X]	1	1	None
IFEQ	A, #	2	2	None
IFEQ	A, M	2	2	None
IFEQ	M, #	3	3	None
IFEQ	X, #	3	3	None
IFGT	A, [#X]	2	3	None
IFGT	A, [X]	1	1	None
IFGT	A, #	2	2	None
IFGT	A, M	2	2	None
IFGT	X, #	3	3	None
IFLT	X, #	3	3	None
IFNBIT	#, A	1	1	None
IFNBIT	#, M	2	2	None
IFNBIT	#, [X]	1	1	None
IFNC		1	1	None
IFNE	A, [#X]	2	3	None
IFNE	A, [X]	1	1	None
IFNE	A, #	2	2	None
IFNE	A, M	2	2	None
IFNE	X, #	3	3	None
IFNE	M, #	3	3	None
INC	A	1	1	Z,N
INC	M	2	2	Z,N

Mnemonic	Operand	Bytes	Cycles	Flags affected
INC	X	1	1	Z
INTR		1	5	None
INVC		1	1	C
JMP	M	3	4	None
JMP	[#, X]	2	3	None
JP		1	1	None
JSR	M	3	5	None
JSR	[#, X]	2	5	None
LD	A, #	2	2	None
LD	A, [#X]	2	3	None
LD	A, [X]	1	1	None
LD	A, M	2	2	None
LD	M, #	3	3	None
LD	M, M	3	3	None
LD	X, #	3	3	None
LDC	#, M	2	2	C
NOP		1	1	None
OR	A, [X]	1	1	Z, N
OR	A, [#X]	2	3	Z,N
OR	A, M	2	2	Z,N
OR	A, #	2	2	Z,N
RBIT	#, [X]	1	2	Z,N
RBIT	#, M	2	2	Z,N
RC		1	1	C,H
RET		1	5	None
RETI		1	5	None
RLC	A	1	1	C,Z,N
RLC	M	2	2	C,Z,N
RRC	A	1	1	C,Z,N
RRC	M	2	2	C,Z,N
SBIT	#, [X]	1	2	Z,N
SBIT	#, M	2	2	Z,N
SC		1	1	C,H
ST	A, [#X]	2	3	None
ST	A, [X]	1	1	None
ST	A, M	2	2	None
STC	#, M	2	2	Z,N
SUBC	A, [X]	1	1	C,H,Z,N
SUBC	A, [#X]	2	3	C,H,Z,N
SUBC	A, M	2	2	C,H,Z,N
SUBC	A, #	2	2	C,H,Z,N
XOR	A, [X]	1	1	Z,N
XOR	A, [#X]	2	3	Z,N
XOR	A, M	2	2	Z,N
XOR	A, #	2	2	Z,N

3.4 Memory Map

All I/O ports, peripheral registers, and core registers (except the accumulator and the program counter) are mapped into the memory space.

Table 10. Memory Mapped Registers

Address	Memory Space	Block	Contents
0x00 - 0x3F	Data	SRAM	Data RAM
0x40 - 0x7F	Data	EEPROM	Data EEPROM
0x80-0x9F	Data	Reserved	
0xA0	Data	HBC	HBCNTRL register
0xA1	Data	HBC	PSCALE register
0xA2	Data	HBC	HPATTERN register
0xA3	Data	HBC	LPATTERN register
0xA4	Data	HBC	BPSEL register
0xA7	Data	Timer1	T1RBLO register
0xA8	Data	Timer1	T1RBHI register
0xA9	Data	HBC	DAT0 register
0xAA	Data	Timer1	T1RALO register
0xAB	Data	Timer1	T1RAHI register
0xAC	Data	Timer1	TMR1LO register
0xAD	Data	Timer1	TMR1HI register
0xAE	Data	Timer1	T1CNTRL register
0xAF	Data	MIW	WKEDG register
0xB0	Data	MIW	WKPND register
0xB1	Data	MIW	WKEN register
0xB2	Data	I/O	PORTGD register
0xB3	Data	I/O	PORTGC register
0xB4	Data	I/O	PORTGP register
0xB5	Data	Timer0	WDSVR register
0xB6	Data	Timer0	T0CNTRL register
0xB7	Data	Clock	HALT mode register
0xB8-0xBA	Data	Reserved	
0xBB	Data	Init. Register	Initialization Register 1
0xBC	Data	Init. Register	Initialization Register 2
0xBD	Data	LBD	LBD register
0xBE	Data	Core	XHI register
0xBF	Data	Core	XLO register
0xC0	Data	Clock	Power Mode Clear (PMC) Register
0xCE	Data	Core	SP register
0xCF	Data	Core	Status register (SR)
0xD0 - 0xFF	Data	Reserved	
0x800 - 0xFF5	Program	EEPROM	Code EEPROM
0xFF6 - 0xFF7	Program	Core	Timer0 Interrupt vector
0xFF8 - 0xFF9	Program	Core	Timer1 Interrupt vector
0xFFA - 0xFFB	Program	Core	MIW Interrupt vector
0xFFC - 0xFFD	Program	Core	Soft Interrupt vector
0xFFE - 0xFFFF	Program	Reserved	

3.5 Memory

The ACEx microcontroller has 64 bytes of SRAM and 64 bytes of EEPROM available for data storage. The device also has 2K bytes of EEPROM for program storage. Software can read and write to SRAM and data EEPROM but can only read from the code EEPROM. While in normal mode, the code EEPROM is protected from any writes. The code EEPROM can only be rewritten when the device is in program mode and if the write disable (WDIS) bit of the initialization register is not set to 1.

While in normal mode, the user can write to the data EEPROM array by 1) polling the ready (R) flag of the SR, then 2) executing the appropriate instruction. If the R flag is 1, the data EEPROM block is ready to perform the next write. If the R flag is 0, the data EEPROM is busy. The data EEPROM array will reset the R flag after the completion of a write cycle. Attempts to

read, write, or enter HALT/IDLE mode while the data EEPROM is busy (R = 0) can affect the current data being written.

3.6 Initialization Registers

The ACEx microcontroller has two 8-bit wide initialization registers. These registers are read from the memory space on power-up to initialize certain on-chip peripherals. Figure 14 provides a detailed description of Initialization Register 1. The Initialization Register 2 is used to trim the internal oscillator to its appropriate frequency. This register is pre-programmed in the factory to yield an internal instruction clock of 1MHz.

The Initialization Registers 1 and 2 can be read from and written to during programming mode. However, re-trimming the internal oscillator (writing to the Initialization Register 2) once it has left the factory is *discouraged*.

Figure 14. Initialization Register 1

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
CMODE[0]	CMODE[1]	WDEN	BOREN	LDBEN	UBD	WDIS	RDIS

- (0) RDIS If set, disables attempts to read the contents from the memory while in programming mode. Once this bit is set, it is no longer possible to unset this option even though the write disable option is not enabled.
- (1) WDIS If set, disables attempts to write new contents to the memory while in programming mode
- (2) UBD If set, the device will not allow any writes to occur in the upper block of data EEPROM (0x60-0x7F)
- (3) LDBEN If set, the Low Battery Detection circuit is enabled
- (4) BOREN If set, allows a BOR to occur if Vcc falls below the voltage reference level
- (5) WDEN If set, enables the on-chip processor watchdog circuit
- (6) CMODE[1] Clock mode select bit 1 (See Table 16)
- (7) CMODE[0] Clock mode select bit 0 (See Table 16)

4. Timer 1

Timer 1 is a versatile 16-bit timer that can operate in one of four modes:

- **Pulse Width Modulation (PWM)** mode, which generates pulses of a specified width and duty cycle
- **External Event Counter** mode, which counts occurrences of an external event
- **Standard Input Capture** mode, which measures the elapsed time between occurrences of external events
- **Difference Input Capture** mode, which automatically measures the difference between edges.

Timer 1 contains a 16-bit timer/counter register (TMR1), a 16-bit auto-reload/capture register (T1RA), a secondary 16-bit auto-reload register (T1RB), and an 8-bit control register (T1CNTRL). All register are memory-mapped for simple access through the core with both the 16-bit registers organized as a pair of 8-bit register bytes {TMR1HI, TMR1LO}, {T1RAHI, T1RALO}, and {T1RBHI, T1RBLO}. Depending on the operating mode, the timer contains an external input or output (T1) that is multiplexed with the I/O pin G2. By default, the TMR1 is reset to 0xFFFF, T1RA/T1RB is reset to 0x0000, and T1CNTRL is reset to 0x00.

The timer can be started or stopped through the T1CNTRL register bit T1C0. When running, the timer counts down (decrements) every clock cycle. Depending on the operating mode, the timer's clock is either the instruction clock or a transition on the T1 input. In addition, occurrences of timer underflow (transitions from 0x0000 to 0xFFFF/T1RA/T1RB value) can either generate an interrupt and/or toggle the T1 output pin.

Timer 1's interrupt (TMRI1) can be enabled by interrupt enable (T1EN) bit in the T1CNTRL register. When the timer interrupt is enabled, depending on the operating mode, the source of the interrupt is a timer underflow and/or a timer capture.

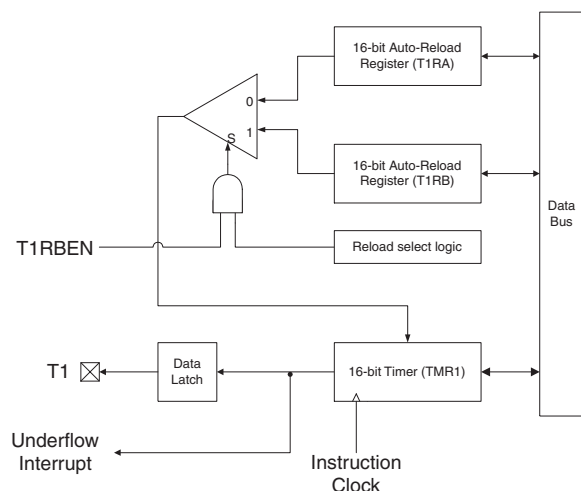
4.1 Timer control bits

Reading and writing to the T1CNTRL register controls the timer's operation. By writing to the control bits, the user can enable or disable the timer interrupts, set the mode of operation, and start or stop the timer. The T1CNTRL register bits are described in Table 11 and Table 12.

The following steps show how to properly configure Timer 1 to operate in the PWM mode. For this example, the T1 output signal is toggled with every timer underflow and the “high” and “low” times for the T1 output can be set to different values. The T1 output signal can start out either high or low depending on the configuration of G2; the instructions below are for starting with the T1 output high. Follow the instructions in parentheses to start the T1 output low.

1. Configure T1 as an output by setting bit 2 of PORTGC.
- SBIT 2, PORTGC ; Configure G2 as an output
2. Initialize T1 to 1 (or 0) by setting (or clearing) bit 2 of PORTGD.
- SBIT 2, PORTGD ; Set G2 high
3. Load the initial PWM high (low) time into the timer register.
- LD TMR1LO, #6FH ; High (Low) for 1.391ms (1MHz clock)
- LD TMR1HI, #05H
4. Load the PWM low (high) time into the T1RA register.
- LD T1RALO, #2FH ; Low (High) for .303ms (1MHz clock)
- LD T1RAHI, #01H
5. Write the appropriate control value to the T1CNTRL register to select PWM mode with T1 toggle, to clear the enable bit and pending flag, and to start the timer. (See Table 11 and Table 12.)
- LD T1CNTRL, #0B0H ; Setting the T1C0 bit starts the timer
6. After every underflow, load T1RA with alternate values. If the user wishes to generate an interrupt on a T1 output transition, reset the pending flags and then enable the interrupt using T1EN. The G bit must also be set. The interrupt service routine must reset the pending flag and perform whatever processing is desired.
- RBIT T1PND, T1CNTRL ; T1PND equals 3
- LD T1RALO, #6FH ; High (Low) for 1.391ms (1MHz clock)
- LD T1RAHI, #05H

Figure 15. Pulse Width Modulation Mode



4.3 Mode 2: External Event Counter Mode

The External Event Counter mode operates similarly to the PWM mode; however, the timer is not clocked by the instruction clock but by transitions of the T1 input signal. The edge is selectable through the T1C1 bit of the T1CNTRL register. A block diagram of the timer's External Event Counter mode of operation is shown in Figure 16.

The T1 input should be connected to an external device that generates a positive/negative-going pulse for each event. By clocking the timer through T1, the number of positive/negative transitions can be counted therefore allowing software to capture the number of events that occur. The input signal on T1 must have a pulse width equal to or greater than one instruction clock cycle.

The counter can be configured to sense either positive-going or negative-going transitions on the T1 pin. The maximum frequency at which transitions can be sensed is one-half the frequency of the instruction clock.

As with the PWM mode, when the counter underflows the counter is reloaded from the T1RA register and the count down proceeds from the loaded value. At every underflow, a pending flag (T1PND) located in the T1CNTRL register is set. Software must then clear the T1PND flag and can then load the T1RA register with an alternate value.

The counter has one interrupt (TMRI1) that is maskable through the T1EN bit of the T1CNTRL register. However, the core is only interrupted if the T1EN bit and the G (Global Interrupt enable) bit of the SR is set. If interrupts are enabled, the counter will generate an interrupt each time the T1PND flag is set (when timer underflows provided that the pending flag was cleared.) The interrupt service routine is responsible for proper handling of the T1PND flag and the T1EN bit.

The following steps show how to properly configure Timer 1 to operate in the External Event Counter mode. For this example, the counter is clocked every falling edge of the T1 input signal. Follow the instructions in parentheses to clock the counter every rising edge.

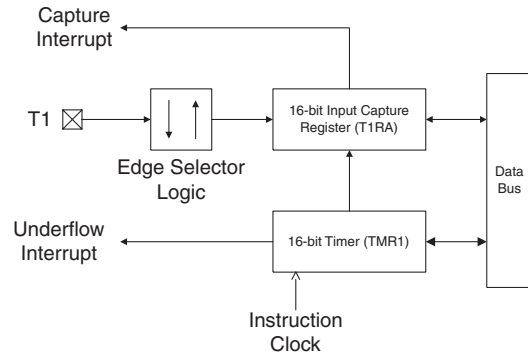
1. Configure T1 as an input by clearing bit 2 of PORTGC.
- RBIT 2, PORTGC ; Configure G2 as an input
2. Initialize T1 to input with pull-up by setting bit 2 of PORTGD.
- SBIT 2, PORTGD ; Set G2 high
3. Enable the global interrupt enable bit.
- SBIT 4, STATUS
4. Load the initial count into the TMR1 and T1RA registers.
When the number of external events is detected, the counter will reach zero; however, it will not underflow until the next event is detected. To count N pulses, load the value N-1 into the registers. If it is only necessary to count the number of occurrences and no action needs to be taken at a particular count, load the value 0xFFFF into the registers.
- LD TMR1LO, #0FFH
- LD TMR1HI, #0FFH
- LD T1RALO, #0FFH
- LD T1RAHI, #0FFH

(See Table 11 and Table 12)

- LD T1CNTRL, #64H ; T1C1 is the edge select bit

- As soon as the input capture mode is enabled, the timer starts counting. When the selected edge is sensed on T1, the T1RA register is loaded and a Timer 1 interrupt is triggered.

Figure 17. Input Capture Mode



4.5 Mode 4: Difference Input Capture Mode

The Difference Input Capture mode works similarly to the standard Input Capture mode. However, for the Difference Input Capture the timer automatically captures the elapsed time between the selected edges without the core needing to perform the calculation.

For example, the standard Input Capture mode requires that the timer be configured to capture a particular edge (rising or falling) at which time the timer's value is copied into the capture register. If the elapsed time is required, software must move the captured data into RAM and reconfigure the Input Capture mode to capture on the next edge (rising or falling). Software must then subtract the difference between the two edges to yield useful information.

The Difference Capture mode eliminates the need for software intervention and allows for capturing very short pulse or cycle widths. It can be configured to capture the elapsed time between:

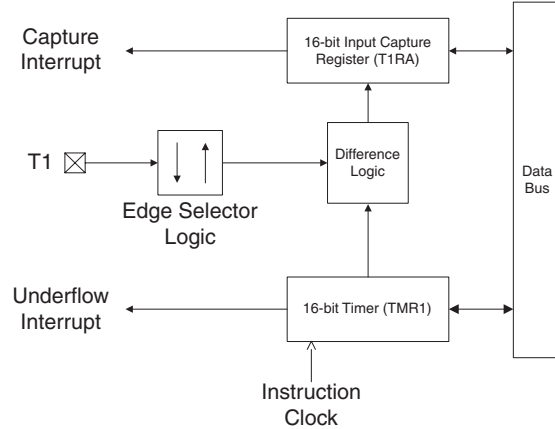
- rising edge to falling edge
- rising edge to rising edge
- falling edge to rising edge
- falling edge to falling edge

Once configured, the Difference Capture timer waits for the first selected edge. When the edge transition has occurred, the 16-bit timer starts counting up based every instruction clock cycle. It will continue to count until the second selected edge transition occurs at which time the timer stops and stores the elapse time into the T1RA register.

Software can now read the difference between transitions directly without using any processor resources. However, like

the standard Input Capture mode both the capture (T1PND) and the underflow (T1C0) flags must be monitored and handled appropriately. This feature allows the ACEx microcontroller to capture very small pulses where standard microcontrollers might have missed cycles due to the limited bandwidth.

Figure 18. Difference Capture Mode



5. Timer 0

Timer 0 is a 12-bit free running idle timer. Upon power-up or any reset, the timer is reset to 0x000 and then counts up continuously based on the instruction clock of 1MHz (1 μs). Software cannot read from or write to this timer. However, software can monitor the timer's pending (TOPND) bit that is set every 8192 cycles (initially 4096 cycles after a reset). The TOPND flag is set every other time the timer overflows (transitions from 0xFFFF to 0x000) through a divide-by-2 circuit. After an overflow, the timer will reset and restart its counting sequence.

Software can either poll the TOPND bit or vector to an interrupt subroutine. In order to interrupt on a TOPND, software must be sure to enable the Timer 0 interrupt enable (TOINTEN) bit in the Timer 0 control (T0CNTRL) register and also make sure the G bit is set in SR. Once the timer interrupt is serviced, software should reset the TOPND bit before exiting the routine. Timer 0 supports the following functions:

- Exiting from IDLE mode (See Section 16 for details.)
- Start up delay from HALT mode
- Watchdog pre-scalar (See Section 6 for details.)

The TOINTEN bit is a read/write bit. If set to 0, interrupt requests from the Timer 0 are ignored. If set to 1, interrupt requests are accepted. Upon reset, the TOINTEN bit is reset to 0.

The TOPND bit is a read/write bit. If set to 1, it indicates that a Timer 0 interrupt is pending. This bit is set by a Timer 0 overflow and is reset by software or system reset.

The WKINTEN bit is used in the Multi-input Wakeup/Interrupt block. See Section 8 for details.

Figure 19. Timer 0 Control Register Definition (T0CNTRL)

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
WKINTEN	x	x	x	x	x	TOPND	TOINTEN

6. Watchdog

The Watchdog timer is used to reset the device and safely recover in the rare event of a processor “runaway condition.” The 12-bit Timer 0 is used as a pre-scalar for Watchdog timer. The Watchdog timer must be serviced before every 61,440 cycles but no sooner than 4096 cycles since the last Watchdog reset. The Watchdog is serviced through software by writing the value 0x1B to the Watchdog Service (WDSVR) register (see Figure 20). The part resets automatically if the Watchdog is serviced too frequent, or not frequent enough.

The Watchdog timer must be enabled through the Watchdog enable bit (WDEN) in the initialization register. The WDEN bit

can only be set while the device is in programming mode. Once set, the Watchdog will always be powered-up enabled. Software cannot disable the Watchdog. The Watchdog timer can only be disabled in programming mode by resetting the WDEN bit as long as the memory write protect (WDIS) feature is not enabled.

WARNING

Ensure that the Watchdog timer has been serviced before entering IDLE mode because it remains operational during this time.

Figure 20. Watchdog Service Register (WDSVR)

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0	0	0	1	1	0	1	1

7. Hardware Bit-Coder

The Hardware Bit-Coder is a dedicated hardware bit-encoding peripheral block, Hardware Bit-Coder (HBC), for IR/RF data transmission (see Figure 21.) The HBC is completely software programmable and can be configured to emulate various bit-encoding formats. The software developer has the freedom to encode each bit of data into a desired pattern and output the encoded data at the desired frequency through either the G2 or G5 output (TX) ports.

The HBC contains six 8-bit memory-mapped configuration registers PSCALE, HPATTERN, LPATTERN, BPSEL, HBCNTRL, and DAT0. The registers are used to select the transmission frequency, store the data bit-encoding patterns, configure the data bit-pattern/frame lengths, and control the data transmission flow.

To select the IR/RF transmission frequency, an 8-bit divide constant must be written into the IR/RF Pre-scalar (PSCALE) register. The IR/RF transmission frequency generator divides the 1MHz instruction clock down by 4 and the PSCALE register is used to select the desired IR/RF frequency shift. Together, the transmission frequency range can be configured between 976Hz (PSCALE = 0xFF) and 125kHz (PSCALE = 0x01). Upon a reset, the PSCALE register is initialized to zero disabling the IR/RF transmission frequency generator. However, once the PSCALE register is programmed, the desired IR/RF frequency is maintained as long as the device is powered.

Once the transmission frequency is selected, the data bit-encoding patterns must be stored in the appropriate registers. The HBC contains two 8-bit bit-encoding pattern registers, High-pattern (HPATTERN) and Low-pattern (LPATTERN). The encoding pattern stored in the HPATTERN register is transmitted when the data bit value to be encoded is a 1. Similarly, the pattern stored in the LPATTERN register is transmitted when the data bit value to be encoded is a 0. The HBC transmits each encoded pattern MSB first.

The number of bits transmitted from the HPATTERN and LPATTERN registers is software programmable through the Bit Period Configuration (BPSEL) register (see Figure 22). During the transmission of HPATTERN, the number of bits transmitted is configured by BPH[2:0] (BPSEL[2:0]) while BPL[2:0] (BPSEL[5:3]) configures the number of transmitted bits for the LPATTERN. The HBC allows from 2 (0x1) to 8 (0x7) encoding

pattern bits to be transmitted from each register. Upon a reset, BPSEL is initially 0 disabling the HBC from transmitting pattern bits from either register.

The Data (DAT0) register is used to store up to 8 bits of data to be encoded and transmitted by the HBC. This data is shifted, bit by bit, MSB to LSB into a 1-bit decision register. If the active bit shifted into the decision register is 1, the pattern in the HPATTERN register is shifted out of the output port. Similarly, if the active bit is 0 the pattern in the LPATTERN register is shifted out.

The HBC control (HBCNTRL) register is used to configure and control the data transmission. HBCNTRL is divided in 5 different controlling signal FRAME[2:0], IOSEL, TXBUSY, START / STOP, and OCFLAG (see Figure 23.)

FRAME[2:0] selects the number of bits of DAT0 to encode and transmit. The HBC allows from 2 (0x1) to 8 (0x7) DAT0 bits to be encoded and transmitted. Upon a reset, FRAME is initialized to zero disabling the DAT0's decision register transmitting no data.

The IOSEL signal selects the transmission to output (TX) through either port G2 or G5. If IOSEL is 1, G5 is selected as the output port otherwise G2 is selected.

The TXBUSY signal is read only and is used to inform software that a transmission is in progress. TXBUSY goes high when the encoded data begins to shift out of the output port and will remain high during each consecutive DAT0 frame bit transmission (see Figure 25). The HBC will clear the TXBUSY signal when the last DAT0 encoded bit of the frame is transmitted and the STOP signal is 0.

The START / STOP signal controls the encoding and transmission process for each data frame. When software sets the START / STOP bit the DAT0 frame transmission process begins. The START signal will remain high until the beginning of the last encoded DAT0 frame bit transmission. The HBC then clears the START / STOP bit allowing software to elect to either continue with a new DAT0 frame transmission or stop the transmission all together (see Figure 25). If TXBUSY is 0 when the START signal is enabled, a synchronization period occurs before any data is transmitted lasting the amount of time to transmit a 0 encoded bit (see Figure 24).

Figure 22. Bit Period Configuration (BPSEL) Register

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0	0	BPL[2:0]			BPH[2:0]		

Figure 23. HBC Control (HBCNTRL) Register

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
OCFLAG	IOSEL	START / STOP	TXBUSY	0	FRAME[2:0]		

Figure 24. HBC signals for one byte message in PWM format

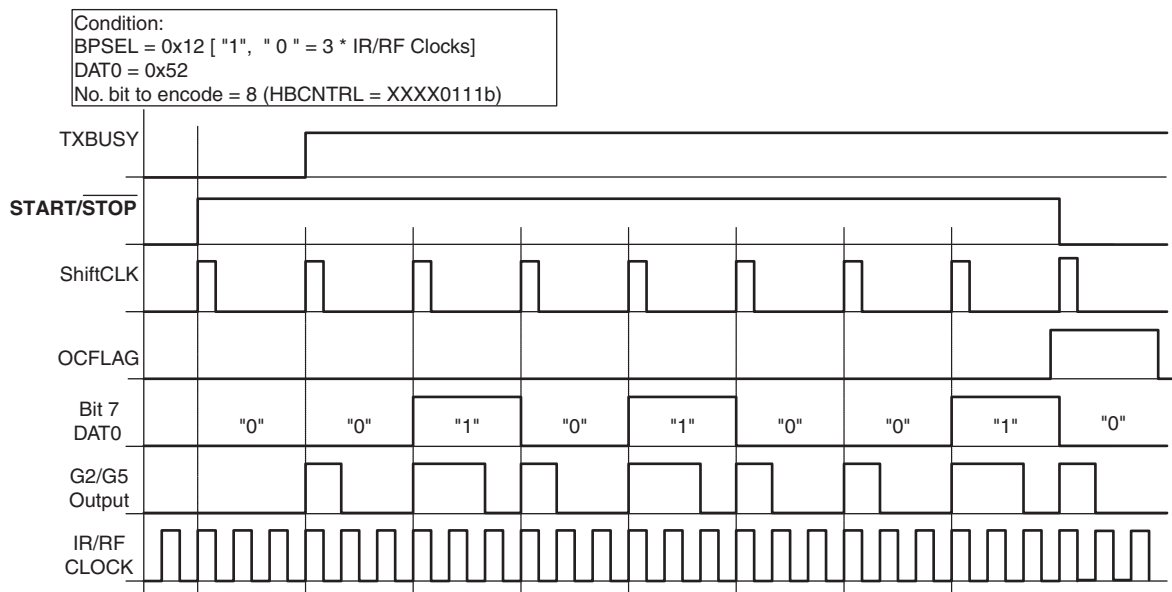


Figure 25. Sending series of encoded messages

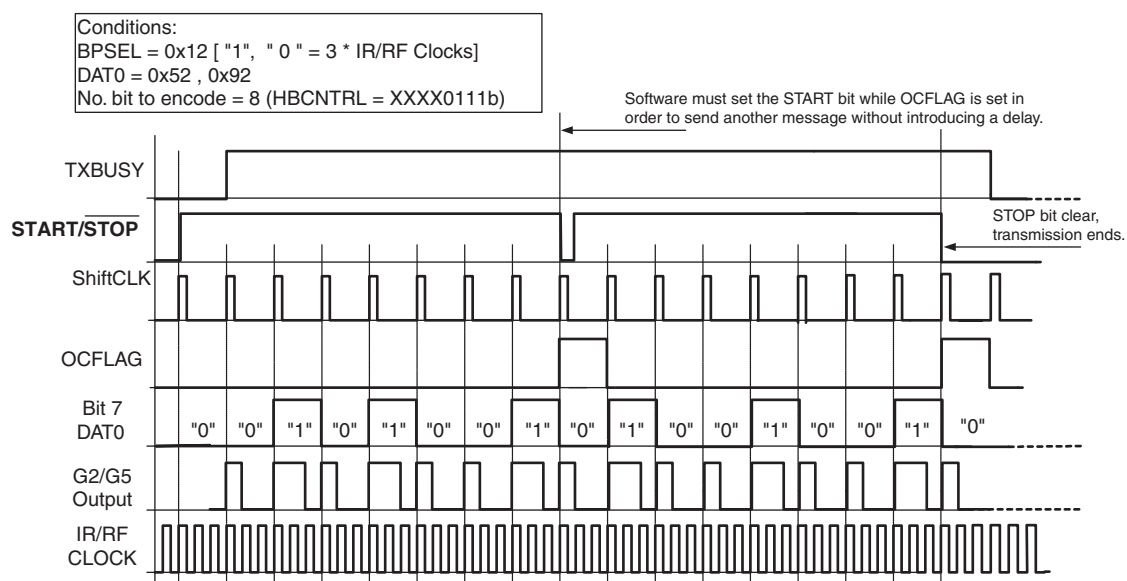
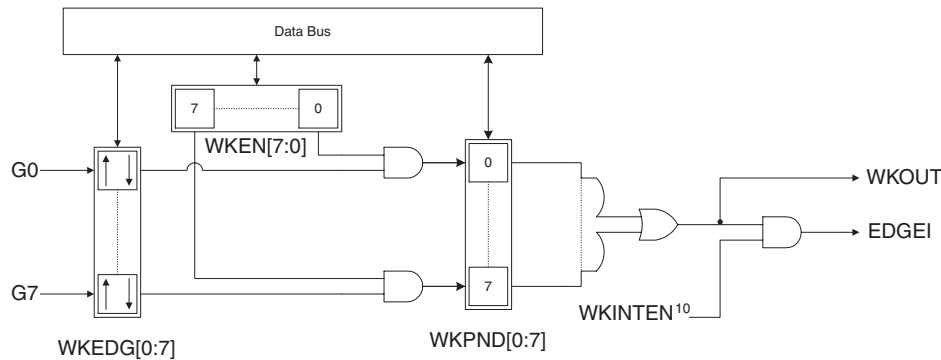


Figure 27. Multi-input Wakeup (MIW) Block Diagram



10. WKINTEN: Bit 7 of T0CTRL

9. I/O Port

The eight I/O pins (six on 8-pin package option) are bi-directional (see Figure 28). The bi-directional I/O pins can be individually configured by software to operate as high-impedance inputs, as inputs with weak pull-up, or as push-pull outputs. The operating state is determined by the contents of the corresponding bits in the data and configuration registers. Each bi-directional I/O pin can be used for general purpose I/O, or in some cases, for a specific alternate function determined by the on-chip hardware.

Figure 28. PORTGD Logic Diagram

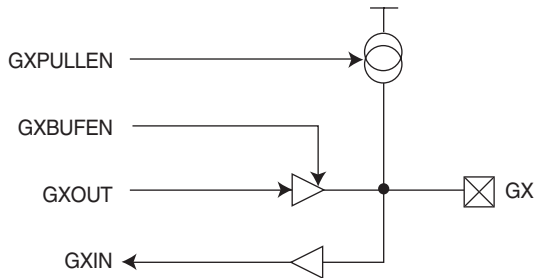


Figure 29. I/O Register bit assignments

PORTGC, PORTGD, PORTGP							
Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
¹¹ G7	¹¹ G6	G5	G4	¹² G3	G2	G1	G0

11. Available only on the 14-pin package option

12. G3 after reset is an input with weak pull-up

Table 13. I/O configuration options

Configuration Bit	Data Bit	Port Pin Configuration
0	0	High-impedance input (TRI-STATE input)
0	1	Input with pull-up (weak one input)
1	0	Push-pull zero output
1	1	Push-pull one output

9.1 I/O registers

The I/O pins (G0-G7) have three memory-mapped port registers associated with the I/O circuitry: a port configuration register (PORTGC), a port data register (PORTGD), and a port input register (PORTGP). PORTGC is used to configure the pins as inputs or outputs. A pin may be configured as an input by writing a 0 or as an output by writing a 1 to its corresponding PORTGC bit. If a pin is configured as an output, its PORTGD bit represents the state of the pin (1 = logic high, 0 = logic low). If the pin is configured as an input, its PORTGD bit selects whether the pin is a weak pull-up or a high-impedance input. Table 13 provides details of the port configuration options. The port configuration and data registers can both be read from or written to. Reading PORTGP returns the value of the port pins regardless of how the pins are configured. Since this device supports MIW, PORTG inputs have Schmitt triggers.

10. In-circuit Programming Specification

The ACEx microcontroller supports in-circuit programming of the internal data EEPROM, code EEPROM, and the initialization registers.

In order to enter into program mode a 10-bit opcode (0x34B) must be shifted into the ACE1502 while the device is executing the internal power on reset (T_{RESET}). The shifting protocol follows the same timing rules as the programming protocol defined in Figure 30.

The opcode is shifted into the ACE1502 serially, MSB first, with the data being valid by the rising edge of the clock. Once the pattern is shifted into the device, the current 10-bit pattern is matched to protocol entrance opcode of 0x34B. If the 10-bit pattern is a match, the device will enable the internal program mode flag so that the device will enter into program mode once reset has completed (see Figure 30.)

The opcode must be shifted in after V_{CC} settles to the nominal level and should end before the power on reset sequence (T_{RESET}) completes; otherwise, the device will start normal execution of the program code. If the external reset is applied by bringing the reset pin low, once the reset pin is release the opcode may now be shifted in and again should end before the reset sequence completes.

10.3 Programming Protocol

After placing the device in program, the programming protocol and commands may be issued.

An externally controlled four-wire interface consisting of a LOAD control pin (G3), a serial data SHIFT-IN input pin (G4), a serial data SHIFT-OUT output pin (G2), and a CLOCK pin (G1) is used to access the on-chip memory locations. Communication between the ACEx microcontroller and the external programmer is made through a 32-bit command and response word described in Table 14. Be sure to either float or tie G5 to V_{CC} for proper programming functionality.

The serial data timing for the four-wire interface is shown in Figure 31 and the programming protocol is shown in Figure 30.

10.3.1 Write Sequence

The external programmer brings the ACEx microcontroller into programming then needs to set the LOAD pin to V_{CC} before shifting in the 32-bit serial command word using the SHIFT_IN and CLOCK signals. By definition, bit 31 of the command word is shifted in first. At the same time, the ACEx microcontroller shifts out the 32-bit serial response to the last command on the

SHIFT_OUT pin. It is recommended that the external programmer samples this signal t_{ACCESS} (500 ns) after the rising edge of the CLOCK signal. The serial response word, sent immediately after entering programming mode, contains indeterminate data.

After 32 bits have been shifted into the device, the external programmer must set the LOAD signal to 0V, and then apply two clock pulses as shown in Figure 30 to complete program cycle.

The SHIFT_OUT pin acts as the handshaking signal between the device and programming hardware once the LOAD signal is brought low. The device sets SHIFT_OUT low by the time the programmer has sent the second rising edge during the LOAD = 0V phase (if the timing specifications in Figure 30 are obeyed).

The device will set the R bit of the Status register when the write operation has completed. The external programmer must wait for the SHIFT_OUT pin to go high before bringing the LOAD signal to V_{CC} to initiate a normal command cycle.

10.3.2 Read Sequence

When reading the device after a write, the external programmer must set the LOAD signal to V_{CC} before it sends the new command word. Next, the 32-bit serial command word (for during a READ) should be shifted into the device using the SHIFT_IN and the CLOCK signals while the data from the previous command is serially shifted out on the SHIFT_OUT pin. After the Read command has been shifted into the device, the external programmer must, once again, set the LOAD signal to 0V and apply two clock pulses as shown in Figure 30 to complete READ cycle. Data from the selected memory location, will be latched into the lower 8 bits of the command word shortly after the second rising edge of the CLOCK signal.

Writing a series of bytes to the device is achieved by sending a series of Write command words while observing the devices handshaking requirements.

Reading a series of bytes from the device is achieved by sending a series of Read command words with the desired addresses in sequence and reading the following response words to verify the correct address and data contents.

The addresses of the data EEPROM and code EEPROM locations are the same as those used in normal operation.

Powering down the device will cause the part to exit programming mode.

Table 14 32-Bit Command and Response Word

Bit Number	Input Command Word	Output Response Word
bits 31-30	Must be set to 0	X
bit 29	Set to 1 to read/write data EEPROM, or the initialization registers, otherwise 0	X
bit 28	Set to 1 to read/write code EEPROM, otherwise 0	X
bits 27-25	Must be set to 0	X
bit 24	Set to 1 to read, 0 to write	X
bits 23-19	Must be set to 0	X
bits 18 -8	Address of the byte to be read or written	Same as Input command word
bits 7-0	Data to be programmed or zero if data is to be read	Programmed data or data read at specified address

11.1 Brown-out Reset

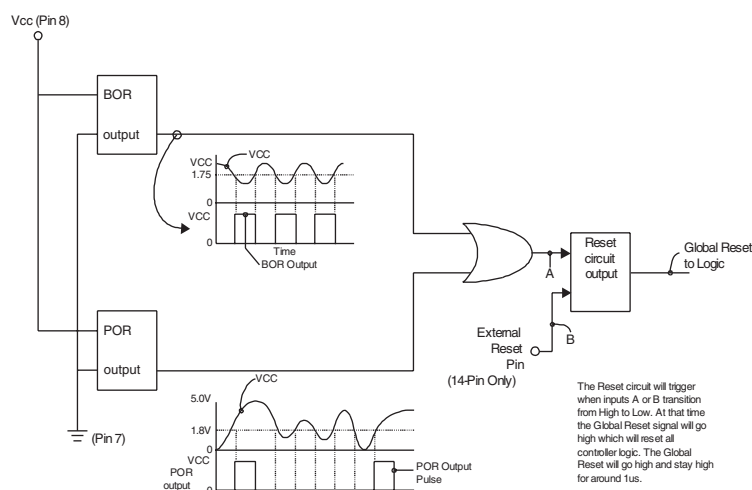
The Brown-out Reset (BOR) function is used to hold the device in reset when Vcc drops below a fixed threshold (1.83V.) While in reset, the device is held in its initial condition until Vcc rises above the threshold value. Shortly after Vcc rises above the threshold value, an internal reset sequence is started. After the reset sequence, the core fetches the first instruction and starts normal operation.

The BOR should be used in situations when Vcc rises and falls slowly and in situations when Vcc does not fall to zero before rising back to operating range. The Brown-out Reset can be thought of as a supplement function to the Power-on Reset if

Vcc does not fall below ~1.5V. The Power-on Reset circuit works best when Vcc starts from zero and rises sharply. In applications where Vcc is not constant, the BOR will give added device stability.

The BOR circuit must be enabled through the BOR enable bit (BOREN) in the initialization register. The BOREN bit can only be set while the device is in programming mode. Once set, the BOR will always be powered-up enabled. Software cannot disable the BOR. The BOR can only be disabled in programming mode by resetting the BOREN bit as long as the global write protect (WDIS) feature is not enabled.

Figure 33. BOR and POR Circuit Relationship Diagram



11.2 Low Battery Detect

The Low Battery Detect (LBD) circuit allows software to monitor the Vcc level at the lower voltage ranges. LBD has a 32-level software programmable voltage reference threshold that can be changed on the fly. Once Vcc falls below the selected threshold, the LBD flag in the LBD control register is set. The LBD flag will hold its value until Vcc rises above the threshold. (See Table 15)

The LBD bit is read only. If LBD is 0, it indicates that the Vcc level is higher than the selected threshold. If LBD is 1, it indicates that the Vcc level is below the selected threshold. The threshold level can be adjusted up to eight levels using the three trim bits (BL[4:0]) of the LBD control register. The LBD flag does not cause any hardware actions or an interruption of the processor. It is for software monitoring only.

The VSEL bit of the LBD control register can be used to select an external voltage source rather than Vcc. If VSEL is 1, the voltage source for the LBD comparator will be an input voltage provided through G4. If VSEL is 0, the voltage source will be Vcc.

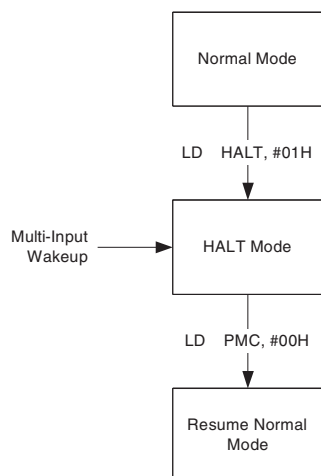
The LBD circuit must be enabled through the LBD enable bit (LBDEN) in the initialization register. The LBDEN bit can only be set while the device is in programming mode. Once set, the LBD will always be powered-up enabled. Software cannot disable the LBD. The LBD can only be disabled in programming mode by

resetting the LBDEN bit as long as the global write protect (WDIS) feature is not enabled.

The LBD circuit is disabled during HALT/IDLE mode. After exiting HALT/IDLE, software must wait at least 10 μ s before reading the LBD bit to ensure that the internal circuit has stabilized.

Table 15. LBD Control Register Definition

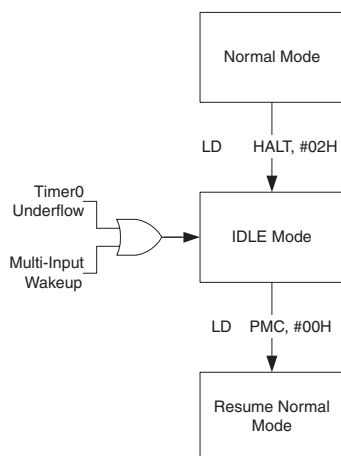
Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
BL[4:0]					VSEL	X	LBD
Level	BL[4]	BL[3]	BL[2]	BL[1]	BL[0]	Voltage Reference Range (Typical)	
1	0	0	0	0	0	1.81V	
2	0	0	0	0	1	1.87V	
3	0	0	0	1	0	1.93V	
4	0	0	0	1	1	1.99V	
5	0	0	1	0	0	2.05V	
6	0	0	1	0	1	2.11V	
7	0	0	1	1	0	2.17V	
8	0	0	1	1	1	2.23V	
9	0	1	0	0	0	2.29V	
10	0	1	0	0	1	2.36V	
11	0	1	0	1	0	2.42V	
12	0	1	0	1	1	2.48V	
13	0	1	1	0	0	2.54V	
14	0	1	1	0	1	2.60V	
15	0	1	1	1	0	2.66V	
16	0	1	1	1	1	2.72V	
17	1	0	0	0	0	2.77V	
18	1	0	0	0	1	2.84V	
19	1	0	0	1	0	2.91V	
20	1	0	0	1	1	2.97V	
21	1	0	1	0	0	3.03V	
22	1	0	1	0	1	3.09V	
23	1	0	1	1	0	3.16V	
24	1	0	1	1	1	3.22V	
25	1	1	0	0	0	3.28V	
26	1	1	0	0	1	3.34V	
27	1	1	0	1	0	3.41V	
28	1	1	0	1	1	3.47V	
29	1	1	1	0	0	3.54V	
30	1	1	1	0	1	3.60V	
31	1	1	1	1	0	3.67V	
32	1	1	1	1	1	3.73V	

Figure 36. Recommended HALT Flow

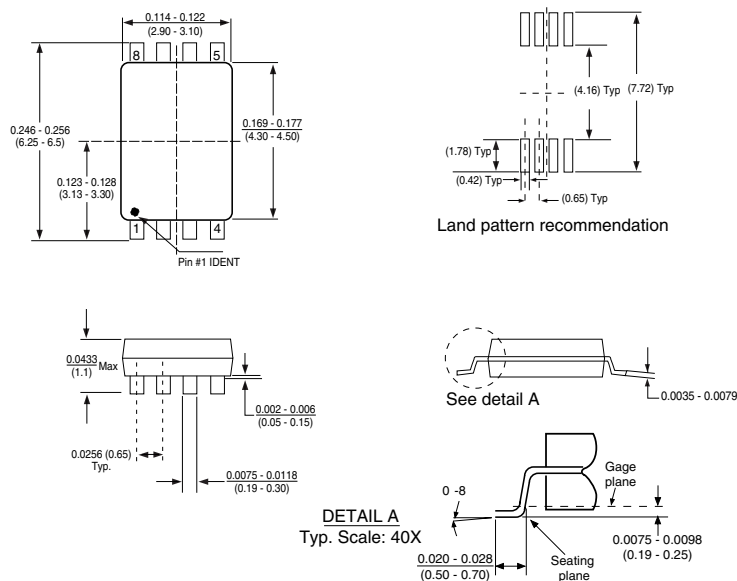
16. IDLE Mode

In addition to the HALT mode power saving feature, the device also supports an IDLE mode operation. The device is placed into IDLE mode by setting the IDLE enable bit (EIDLE) of the HALT register through software using only the “LD M, #” instruction. EIDLE is a write only bit and is automatically cleared upon exiting IDLE. The IDLE mode operation is similar to HALT except the internal oscillator, the Watchdog, and the Timer 0 remain active while the other on-chip systems including the LBD and the BOR circuits are shut down.

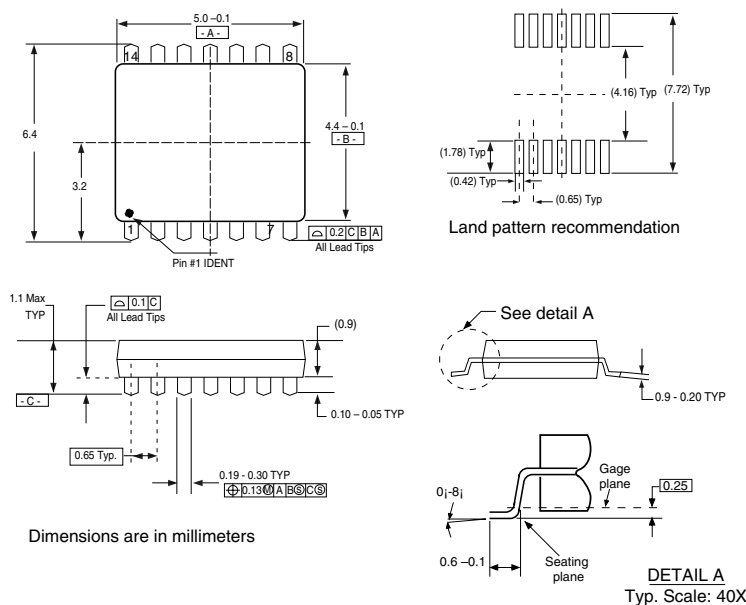
The device automatically wakes from IDLE mode by the Timer 0 overflow every 8192 cycles (see Section 5). Before entering IDLE mode, software must clear the WKEN register to disable the MIW block. Once a wake from IDLE mode is triggered, the core will begin normal operation by the next clock cycle. Immediately after exiting IDLE mode, software must clear the Power Mode Clear (PMC) register by using only the “LD M, #” instruction. (See Figure 37.)

Figure 37. Recommended IDLE Flow

Physical Dimensions inches (millimeters) unless otherwise noted)



8-Pin TSSOP Order Number ACE1502EMT8/ACE1502VMT8 Package Number MT08A



14-Pin TSSOP Order Number ACE1502EMT/ACE1502VMT Package Number MT14A

ACEx Development Tools

General Information:

Fairchild Semiconductor offers different possibilities to evaluate and emulate software written for ACEx.

Simulator: Is a Windows program able to load, assemble, and debug ACEx programs. It is possible to place as many breakpoints as needed, trace the program execution in symbolic format, and program a device with the proper options. The ACEx Simulator is available free-of-charge and can be downloaded from Fairchild's web site at www.fairchildsemi.com/products/memory/ace



ACEx Emulator Kit: Fairchild also offers a low cost real-time in-circuit emulator kit that includes:

- Emulator board
- Emulator software
- Assembler and Manuals
- Power supply
- DIP14 target cable
- PC cable

The ACEx emulator allows for debugging the program code in a symbolic format. It is possible to place one breakpoint and watch various data locations. It also has built-in programming capability.

Prototype Board Kits: Fairchild offers two solutions for the simplification of the breadboard operation so that ACEx Applications can be quickly tested.

- 1) ACEDEMO can be used for general purpose applications
- 2) ACETXRX is for transmitting / receiving (RF, IR, RS232, RS485) applications.

ACEDEMO has 8 switches, 8 LEDs, RS232 voltage translator, buzzer, and a lamp with a small breadboard area.

Factory Programming:

Fairchild offers factory pre-programming and serialization (for justified quantities) for a small additional cost. Please refer to your local distributor for details regarding factory programming.

Ordering P/Ns

Emulator Kit and Programming adapters:

Please refer to your local distributor for details regarding development tools.

Life Support Policy

Fairchild's products are not authorized for use as critical components in life support devices or systems without the express written approval of the President of Fairchild Semiconductor Corporation. As used herein:

1. Life support devices or systems are devices or systems which, (a) are intended for surgical implant into the body, or (b) support or sustain life, and whose failure to perform, when properly used in accordance with instructions for use provided in the labeling, can be reasonably expected to result in a significant injury to the user.
2. A critical component is any component of a life support device or system whose failure to perform can be reasonably expected to cause the failure of the life support device or system, or to affect its safety or effectiveness.

**Fairchild Semiconductor
Americas
Customer Response Center**
Tel: 1-888-522-5372

**Fairchild Semiconductor
Europe**
Deutsch Tel: +49 (0) 8141-6102-0
English Tel: +44 (0) 1793-856856
Français Tel: +33 (0) 1-6930-3696
Italiano Tel: +39 (0) 2-249111-1

**Fairchild Semiconductor
Hong Kong**
8/F, Room 808, Empire Centre
68 Mody Road, Tsimshatsui East
Kowloon, Hong Kong
Tel: +852-2722-8338
Fax: +852-2722-8383

**Fairchild Semiconductor
Japan Ltd.**
4F, Natsume Bldg.
2-18-6, Yushima, Bunkyo-ku
Tokyo, 113-0034 Japan
Tel: 81-3-3818-8840
Fax: 81-3-3818-8841