September 2003

ACE2202 Product Family Arithmetic Controller Engine (ACEx™) for Low Power Applications

General Description

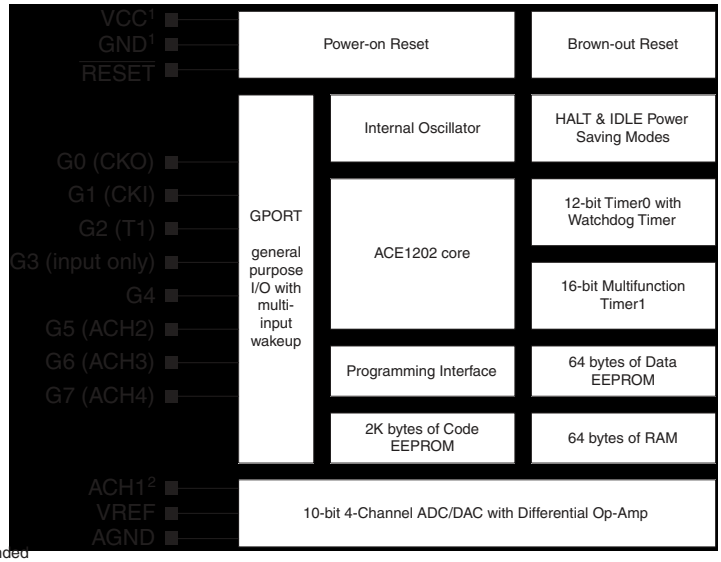
The ACE2202 (Arithmetic Controller Engine) is a member of the ACEx microcontroller family. It is a dedicated programmable monolithic integrated circuit for applications requiring high performance, low power, and small size. It is a fully static part fabricated using CMOS technology.

The ACE2202 product family has an 8-bit microcontroller core, 64 bytes of RAM, 64 bytes of data EEPROM and 2K bytes of code EEPROM. Its on-chip peripherals include a multi-function 16-bit timer, watchdog/idle timer, and programmable undervoltage detection circuitry. The on-chip clock and reset functions reduce the number of required external components. The ACE2202 product family is available in 8- and 14-pin SOIC packages.

Features

- Arithmetic Controller Engine
- 2K bytes on-board code EEPROM
- 64 bytes data EEPROM
- 64 bytes RAM
- Instruction set geared for block encryption
- On-chip oscillator
  - — No external components
  - — 1µs instruction cycle time
- On-chip Power-on Reset
- Brown-out Reset
- 10-bit Analog-to-Digital Converter (ADC)
  - — Differential Operational Amplifier with a gain of 18
  - — ±3 LSB Accuracy
  - — 4 Multiplexed Input channels
  - — Rail-to-Rail input range
- 10-bit Digital-to-Analog Converter (DAC)
  - — ±3 LSB Accuracy
  - — 4 Multiplexed output channels
  - — Rail-to-Rail output range
- 16-bit multifunction timer with difference capture
- 12-bit idle timer with Watchdog
- Memory mapped I/O
- Software selectable I/O option
  - — Push-pull outputs with tri-state option
  - — Weak pull-up or high impedance
- Multi-input wake-up on all eight general purpose I/O pins
- Fully static CMOS
  - — Low power HALT mode
- Single supply operation
  - — 2.4-5.5V (ACE2202)
- Programmable read and write disable functions
- 40 years data retention
- 1,000,000 data changes
- 14-pin TSSOP packages
- In-circuit programming

Block and Connection Diagram

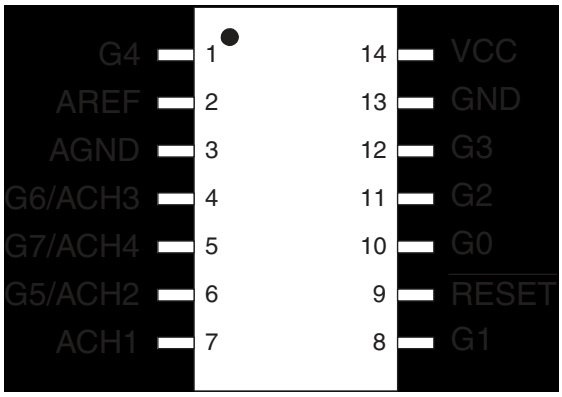


[1]100nf Decoupling capacitor recommended
[2]Only a dedicated analog input

©2003 Fairchild Semiconductor Corporation

Figure 2. ACE2202 TSSOP 14-pin Device Pinout

a) Normal Mode Operation



b) Programming Mode Operation

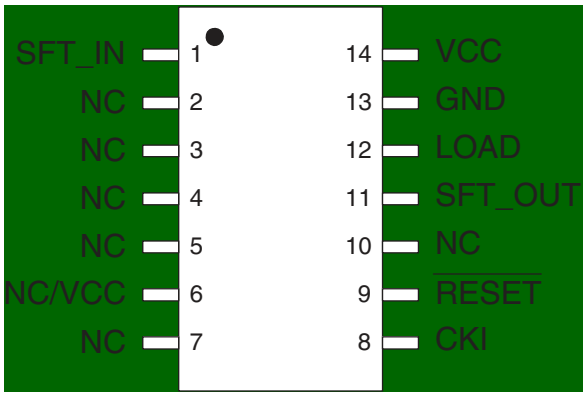
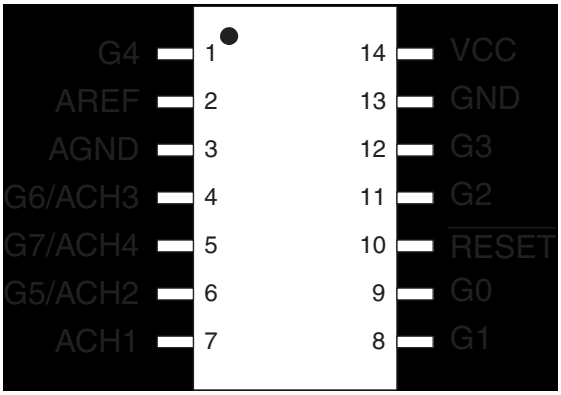
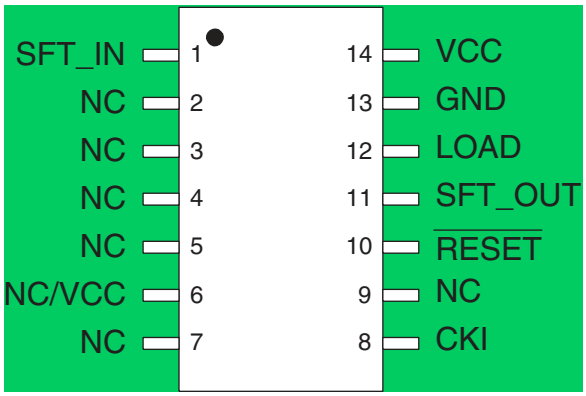


Figure 3. ACE2202 DIP 14-pin Device Pinout

a) Normal Mode Operation



b) Programming Mode Operation



Note: DIP-14 version available for samples only.

2.0 Electrical Characteristics

All characterization data shows in advanced information and has been collected from only one wafer lot.

Absolute Maximum Ratings

Ambient Storage Temperature	-65°C to +150°C
Input Voltage not including G3	-0.3V to V_{CC}+0.3V
G3 Input Voltage	0.3V to 13V
Lead Temperature (10s max)	+300°C
Electrostatic Discharge on all pins	2000V min

Operating Conditions

Relative Humidity (non-condensing)	95%
EEPROM write limits	See DC Electrical Characteristics

Device Number	Operating Voltage	Ambient Operating Temperature
ACE2202	2.4 to 5.5V	0°C to 70°C

Advanced ACE2202 DC Electrical Characteristics

V_{CC} = 2.4 to 5.5V

All measurements valid for ambient operating temperature range unless otherwise stated.

Symbol	Parameter	Conditions	Min	Typ	Max	Units
I_{CC}[3]	Supply Current – no data EEPROM write in progress	2.7V 3.3V 5.5V		0.7 1.2 3.7	1.2 2.0 5.5	mA mA mA
I_{CCH}	HALT Mode current	5.5V @ 0°C to +25°C			30	µA
V_{CCW}	EEPROM Write Voltage	Code EEPROM in Programming Mode	4.5	5.0	5.5	V
		Data EEPROM in Operating Mode	2.4		5.5	V
S_{VCC}	Power Supply Slope		1µs/V		10ms/V	
V_{IL}	Input Low with Schmitt Trigger Buffer	V_{CC} = 2.7 -5.5V			$0.2V_{CC}$	V
V_{IH}	Input High with Schmitt Triggr Buffer	V_{CC} = 2.7 - 5.5V	$0.8V_{CC}$			V
I_{IP}	Input Pull-up Current	V_{CC} = 5.5V, V_{IN} = 0V	30	65	350	µA
I_{TL}[5]	TRI-STATE Leakage, G0 – G6	V_{CC} = 5.5V		2	200	nA
V_{OL}	Output Low Voltage:	V_{CC} = 2.4V – 3.3V				
	G0, G1, G2, G4, G6, G7	2.0 mA sink			$0.2V_{CC}$	V
	G5	5.0 mA sink			$0.2V_{CC}$	V
	Output Low Voltage:	V_{CC} = 3.3V – 5.5V				
	G0, G1, G2, G4, G6, G7	5.0 mA sink			$0.2V_{CC}$	V
	G5	10.0 mA sink			$0.2V_{CC}$	V
V_{OH}	Output High Voltage:	V_{CC} = 2.4V – 3.3V				
	G0, G1, G2, G4, G6, G7	0.4 mA source	$0.8V_{CC}$			V
	G5	0.8 mA source	$0.8V_{CC}$			V
	Output High Voltage:	V_{CC} = 3.3V – 5.5V				
	G0, G1, G2, G4, G6, G7	0.4 mA source	$0.8V_{CC}$			V
	G5	1.0 mA source	$0.8V_{CC}$			V

[3] I_{CC} active current is dependent on the program code.

[4] The parameter is guaranteed by design but not 100% tested.

[5] See section 5.3 for recommendation on port G7.

Advanced ACE2202 DC Electrical Characteristics

V_{CC} = 2.4 to 5.5V

All measurements valid for ambient operating temperature range unless otherwise stated.

Parameter	Conditions	Min	Typ	Max	Units
Instruction cycle time from internal clock - setpoint	5.0V at +25°C	0.9	1.0	1.1	µs
Internal clock voltage dependent frequency variation	3.0V to 5.5V, constant temperature			+6	%
Internal clock temperature dependent frequency variation	3.0V to 5.5V, full temperature range			+8	%
Crystal oscillator frequency	(Note 6)			4	MHz
External clock frequency	(Note 7)			4	MHz
EEPROM write time			3	10	ms
Internal clock start up time	(Note 7)			2	ms
Oscillator start up time	(Note 7)			2400	cycles

[6]The maximum permissible frequency is guaranteed by design but not 100% tested.

[7]The parameter is guaranteed by design but not 100% tested.

Advanced ACE2202 DC Electrical Characteristics for programming

All data following is valid between 4.5V and 5.5V at ambient temperature. The following characteristics are guaranteed by design but are not 100% tested. See "EEPROM write time" in the AC Electrical Characteristics for definition of the programming ready time.

Parameter	Description	Min	Max	Units
t_{HI}	CLOCK high time	500	DC	ns
t_{LO}	CLOCK low time	500	DC	ns
t_{DIS}	SHIFT_IN setup time	100		ns
t_{DIH}	SHIFT_IN hold time	100		ns
t_{DOS}	SHIFT_OUT setup time	100		ns
t_{DOH}	SHIFT_OUT hold time	900		ns
t_{SV1}, t_{SV2}	LOAD supervoltage timing	50		µs
t_{LOAD1}, t_{LOAD2}, t_{LOAD3}, t_{LOAD4}	LOAD timing	5		µs
$V_{SUPERVOLTAGE}$	Supervoltage level	11.5	12.5	V

Advanced ACE2202 Brown-out Reset (BOR) Characteristics

V_{CC} = 2.4 to 5.5V

The following characteristics are guaranteed by design but are not 100% tested.

Parameter	Conditions	Min	Typ	Max	Units
BOR Trigger Threshold	0°C		2.19		V
	+25°C		2.27		V
	+70°C		2.38		V

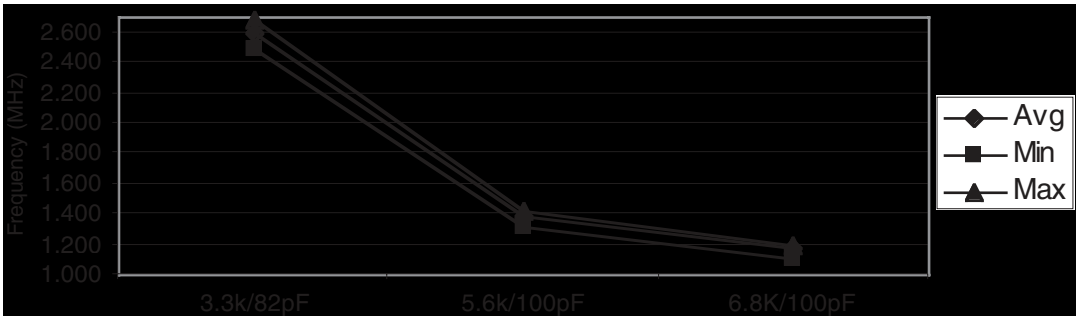
ACE2202 A/D Specifications V_{CC} = 5V, 0°C to 70°C

Parameter	Conditions	Min	Typ	Max	Units
Resolution				10	Bits
Conversion Time			250		uS
Reference Voltage Input	AGND = 0	2.5		V_{CC}	V
Absolute Accuracy	ACH1, ACH2, ACH3 AREF = V_{CC} ACH4 – ACH1 AREF = V_{CC} After Offset cancellation (Differential channel ACH4-ACH1) and curve fitting (ACH1 to ACH4)		±3 ±3		LSB
Non_Linearity	ACH1, ACH2, ACH3 AREF = V_{CC} ACH4 – ACH1 AREF = V_{CC}		±3 ±3		LSB
Differential Non-Linearity	ACH1, ACH2, ACH3 AREF = V_{CC} ACH4 – ACH1 AREF = V_{CC}		±3		LSB
Input Reference Resistance			120		ohms
Difference Amplifier Gain			18		
Difference Amplifier Offset Voltage					V
Sample & Hold Capacitance			32		pF

3.0 AC & DC Electrical Characteristic Graphs

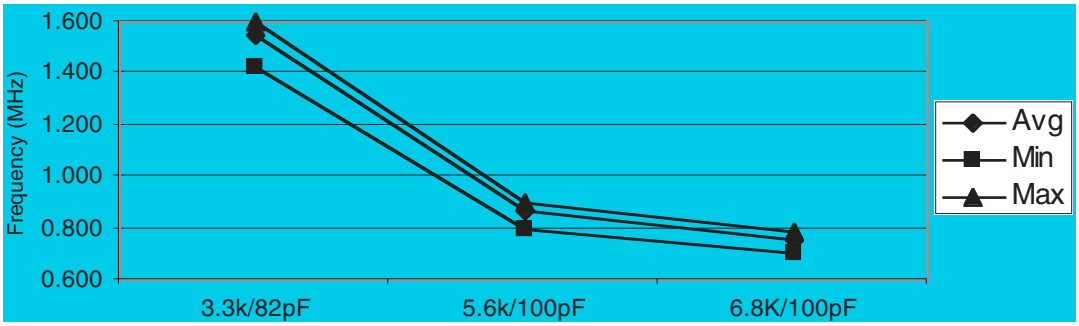
The graphs in this section are for design guidance and are based on preliminary test data

Figure 4. RC Oscillator Frequency vs. Temperature (V_{CC} = 5.0V)



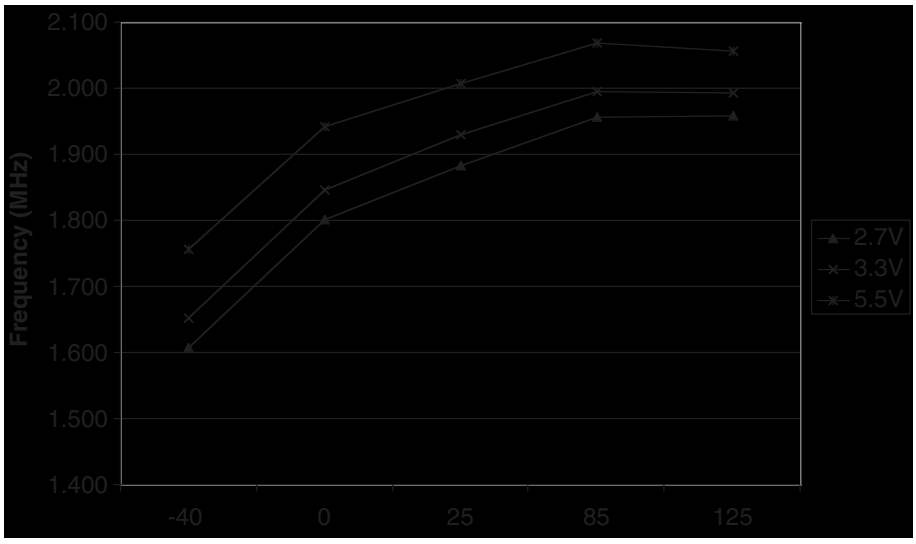
Resistor & Capacitor Values [k & pF]

Figure 5. RC Oscillator Frequency vs. Temperature (V_{CC} = 2.5V)



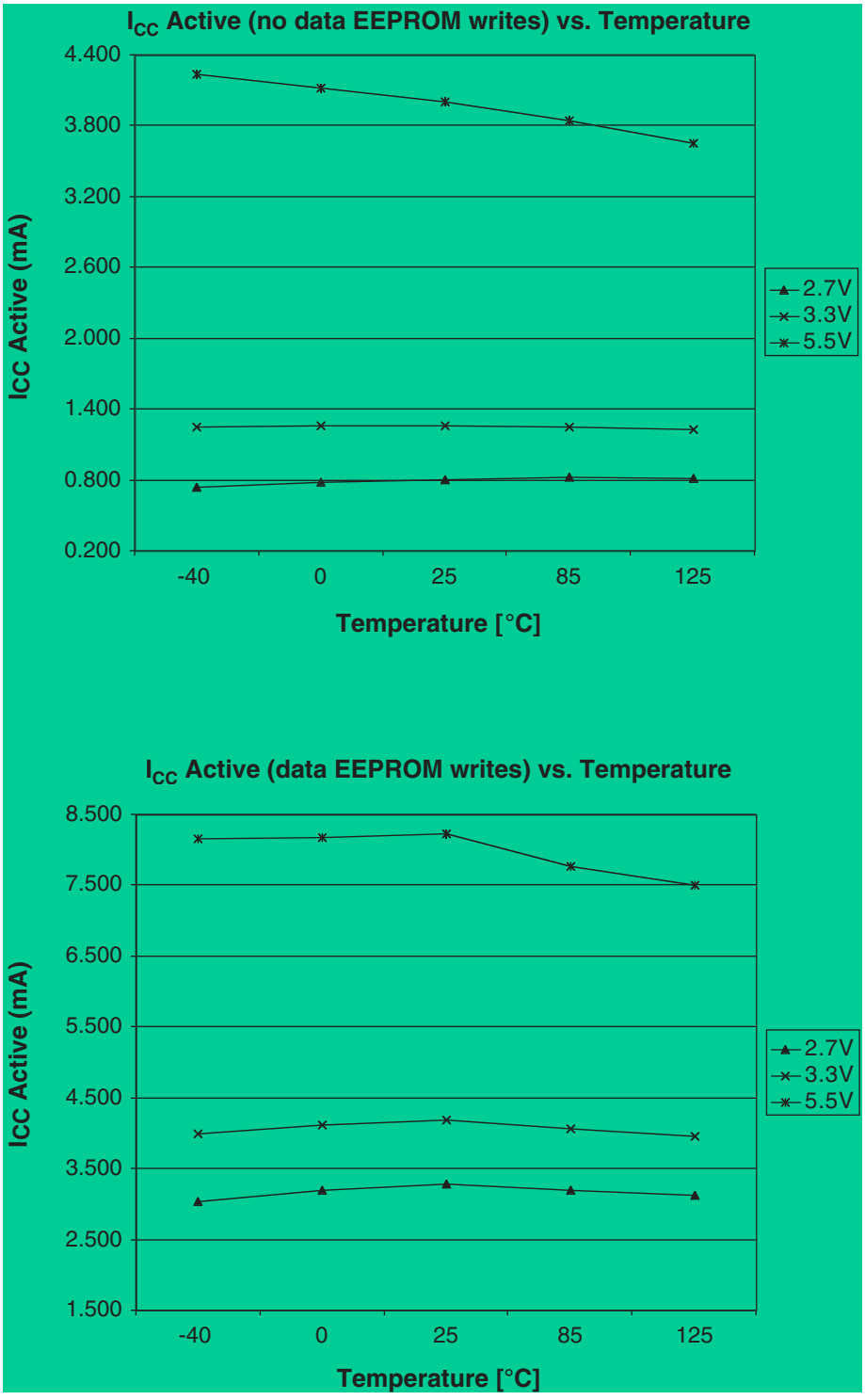
Resistor & Capacitor Values [k & pF]

Figure 6. Internal Oscillator Frequency



Temperature [°C]

Figure 7. I_{CC} Active



4.0 Arithmetic Controller Core

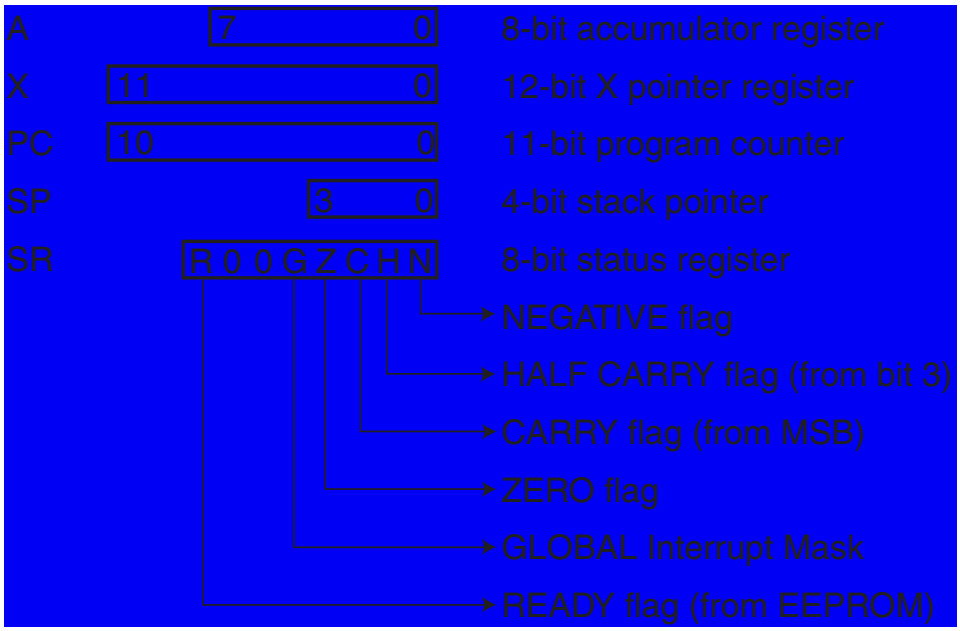
The ACEx microcontroller core is specifically designed for low cost applications involving bit manipulation, shifting and block encryption. It is based on a modified Harvard architecture meaning peripheral, I/O, and RAM locations are addressed separately from instruction data.

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 ACEx microcontroller and takes advantage of the flexibility found on Von Neumann style machines.

4.1 CPU Registers

The ACEx 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 8. Programming Model



4.1.1 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.

4.1.2 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 ACEx 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 7) However, software may also access the reigster 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 9)

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 0xFFF) 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 0xFFF 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 12^{th} bit is dedicated for memory space selection.

4.1.3 Program Counter (PC)

The 10-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.

4.1.4 Stack Pointer (SP)

The ACEx 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 is 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.

4.1.5 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). (See Section 4.5) The condition codes are automatically updated by most instructions. The interrupt masking bit (G) is the only writable flag in the Status register all the other bits are read-only. For additional information see the "ACEx Device Help" guide available in the ACEx development tool suite.

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 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.

4.2 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 9). 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 five interrupts. Four are maskable through the G bit of the SR and the fifth (software interrupt) is not inhibited by the G bit (see Figure 10). The software interrupt instruction 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 in shown in Table 6.

Table 6. Interrupt Priority Sequence

Priority (5 highest, 1 lowest)	Interrupt
5	MIW (EDGEI)
4	Timer0 (TMRI0)
3	Timer1 (TMRI1)
2	ADC (ADC1)
1	Software (INTR)

4.3 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 a 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.

Figure 9. Basic Interrupt Structure

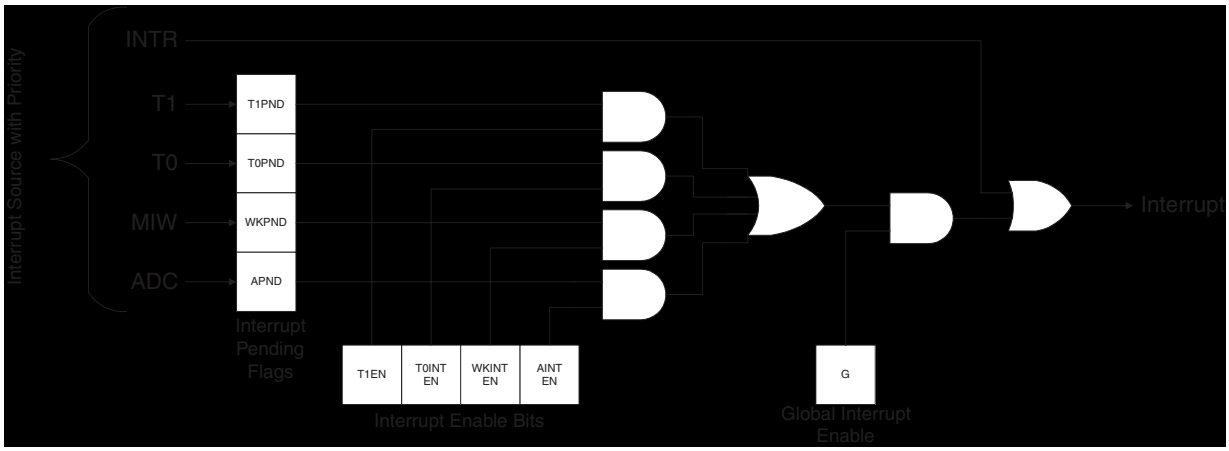


Table 7. Instruction Addressing Modes

Instruction	Immediate			Direct		Indexed	Indirect	Inherent		Relative	Absolute
ADC	A, #			A, M			A, [X]				
ADD	A, #			A, M			A, [X]				
AND	A, #			A, M			A, [X]				
OR	A, #			A, M			A, [X]				
SUBC	A, #			A, M			A, [X]				
XOR	A, #			A, M			A, [X]				
CLR				M				A	X		
INC				M				A	X		
DEC				M				A	X		
IFEQ	A, #	X, #	M,#	A, M		A, [00,X]	A, [X]				
IFGT	A, #	X, #		A, M		A, [00,X]	A, [X]				
IFNE	A, #			A, M		A, [00,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, #	X, #	M, #	A, M	M, M	A, [00,X]	A, [X]				
ST				A, M		A, [00,X]	A, [X]				
NOP								no-op			
IFBIT	#, A			#, M							
SBIT				#, M			#, [X]				
RBIT				#, M			#, [X]				
JP										Rel	
JSR						[00,X]					M
JMP						[00,X]					M
RET								no-op			
RETI								no-op			
INTR								no-op			

Table 8. Instruction Cycles and Bytes

Mnemonic	Operand	Bytes	Cycles	Flags affected
ADC	A, [X]	1	1	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, M	2	2	Z,N
ADD	A, #	2	2	Z,N
AND	A, #	2	2	Z,N
AND	A, M	2	2	Z,N
AND	A, [X]	1	1	Z,N
CLR	X	1	1	Z
CLR	A	1	1	C,H,Z,N
CLR	M	2	2	C,H,Z,N
DEC	A	1	1	Z,N
DEC	M	2	2	Z,N
DEC	X	1	1	Z
IFBIT	#, A	1	1	None
IFBIT	#, M	2	2	None
IFC		1	1	None
IFEQ	A, [00,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, #	2	2	None
IFGT	A, [00,X]	2	3	None
IFGT	A, [X]	1	1	None
IFGT	A, M	2	2	None
IFGT	X, #	3	3	None
IFNE	A, #	2	2	None
IFNE	A, [00,X]	2	3	None
IFNE	A, [X]	1	1	None
IFNE	A, M	2	2	None
IFLT	X, #	3	3	None
IFNC		1	1	None
INC	A	1	1	Z,N
INC	M	2	2	Z,N
INC	X	1	1	Z
INTR		1	5	None
INVC		1	1	C

Mnemonic	Operand	Bytes	Cycles	Flags affected
JMP	M	3	4	None
JMP	[00,X]	2	3	None
JP		1	1	None
JSR	M	3	5	None
JSR	[00,X]	2	5	None
LD	A, #	2	2	None
LD	A, [00,X]	2	3	None
LD	A, [X]	1	1	None
LD	A, M	2	2	None
LD	M, #	3	3	None
LD	X, #	3	3	None
LDC	#, M	2	2	C
LD	M, M	3	3	None
NOP		1	1	None
OR	A, #	2	2	Z,N
OR	A, [X]	1	1	Z,N
OR	A, M	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, [00,X]	2	3	None
ST	A, [X]	1	1	None
ST	A, M	2	2	None
STC	#, M	2	2	Z,N
SUBC	A, #	2	2	C,H,Z,N
SUBC	A, [X]	1	1	C,H,Z,N
SUBC	A, M	2	2	C,H,Z,N
XOR	A, #	2	2	Z,N
XOR	A, [X]	1	1	Z,N
XOR	A, M	2	2	Z,N

4.4 Memory Map

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

Table 9. Memory Map

Address	Memory Space	Block	Contents
0x00 - 0x3F	Data	SRAM	Data RAM
0x40 - 0x7F	Data	EEPROM	Data EEPROM
0x80 - 0x9C			Reserved
0x9D	Data	ADDA	ADDALO register
0x9E	Data	ADDA	ADDAHI register
0x9F	Data	ADDA	ADCNTRL register
0xA0 - 0xA9			Reserved
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 - 0xBD			Reserved
0xBE	Data	Core	XHI register
0xBF	Data	Core	XLO register
0xC0	Data	Core	Power mode clear (PMC) register
0xCE	Data	Core	SP register
0xCF	Data	Core	Status register (SR)
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	Software Interrupt vector
0xFFE - 0xFFF	Program	Core	ADC Interrupt vector

4.5 Memory

The ACEx microcontroller device 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.

4.6 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 10 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.

Both 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 10. 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	BLSEL[10]	UBD[8,9]	WDIS[8,9]	RDIS[8,9]

(0) RDIS[8,9] — If set, disables attempts to read the contents from the memory while in programming mode

(1) WDIS[8,9] — If set, disables attempts to write new contents to the memory while in programming mode

(2) UBD[8,9] — If set, the device will not allow any writes to occur in the upper block of data EEPROM (0x60-0x7F)

(3) BLSEL[10] — BLSEL is the Brown-out Reset (BOR) voltage reference level. It is set to “1” in the factory.

(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 15)

(7) CMODE[0] — Clock mode select bit 0 (See Table 15)

[8]If both the WDIS and RDIS bits are set, the device will no longer be able to be placed into program mode.

[9]If the RDIS or UBD bits are not set while the WDIS bit is not set, then the RDIS and UBD bits can be reset.

[10]The BLSEL bit is set to its appropriate level in the factory. If writing to the initialization register is necessary, be sure to maintain BLSEL set value.

5.0 Analog-to-Digital/Digital-to-Analog (ADDA) Converter

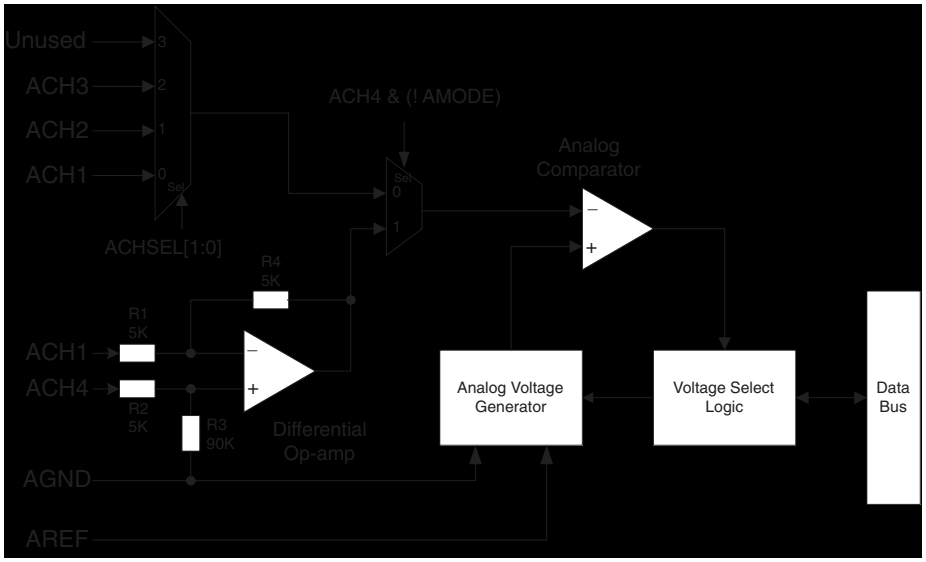
The ADDA block extends the features of the ACEx microcontroller by offering a 4-channel Analog-to-Digital (ADC) with a differential operational amplifier (op-amp) and a 10-bit Digital-to-Analog Converter (DAC). (See Figure 11) The integrated ADC/DAC function offers a single cost-effective solution for applications requiring temperature, voltage, and current sensing.

The ADDA block contains 4 software selectable I/O channels where 3 are multiplexed with ACEx's general purpose I/Os and 1 is dedicated. It has an independent external voltage reference (VREF) as well as an analog ground (AGND) for higher accuracy and noise reduction. The voltage at VREF must be less than or equal to V_{CC}.

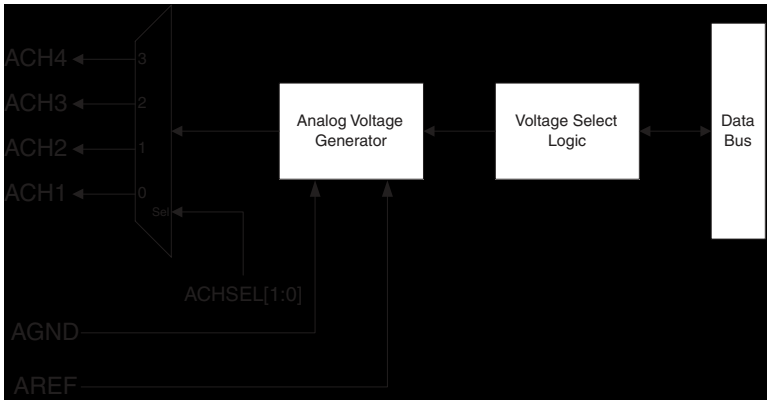
The ADC/DAC performs rail-to-rail (0-VREF) conversions of 10-bit resolution with an accuracy of ±2.5 LSB. When performing an ADC with the use of the differential op-amp, voltage differences on ACH4 and ACH1 can be converted with a gain of 18. The ADC has a conversion time of approximately 240µs while the DAC is less than 10µs. The ADDA block offers software the use of interrupts, which occurs once an ADC process is complete, so that core is freed to perform other tasks.

Figure 11. ADC/DAC Block Diagram

(a) ADC



(b) DAC



5.1 ADDA Control Registers

The ADDA block features are controlled and configured through three memory-mapped registers (ADDALO, ADDAHI, and ADCNTRL) that are accessible to software through the ACEx core. After a reset, the registers are all initialized to 0x00.

ADDALO and ADDAHI are multiplexed registers used by software to read/write the 10-bit digital word for the ADC/DAC. ADDALO represents the lower 8-bits of the 10-bit digital word while ADDAHI represents the upper 2-bits.

Software has both read and write access to the ADDALO and ADDAHI registers. Since the ADDALO and ADDAHI registers are multiplexed, if writing to the registers the data is always written to the DAC specific register. However if the ADDA is configured for an ADC operation the digital converted word is read; otherwise, the digital word to be converted during a DAC operation is read.

The 8-bit ADCNTRL register contains all the signals used to control and configure the ADDA block. (See Figure 13)

The ACHSEL signal is a 2-bit configuration value that allows the user to select between the four analog I/O (ACH1, ACH2, ACH3, or ACH4). (See Table 10)

The AMODE bit is used to configure the ADDA to perform either an ADC or a DAC. If AMODE is '0,' the block will be configured for an ADC making the selected channel (selected by ACHSEL) an input. If AMODE is '1,' the block will be configured for a DAC making the selected channel an output.

Setting AMODE to '0' and selecting channel 4 configures the ADDA block to the differential op-amp mode setting both ACH4 and ACH1 as inputs. ACH4 becomes the positive terminal of the op-amp while ACH1 is the negative.

ADDA has the ability through the AINTEN bit to cause an interrupt of the ACEx core. If AINTEN is '1,' the ADDA Interrupt (ADDAI) is enabled otherwise it is disabled. However, an interrupt will not occur unless the Global Interrupt (G) enable bit in the Status Register (SR) is enabled. (See Section 4.1)

The APND bit is the key controlling bit for the ADDA block. APND is used as a flag to instruct software that the ADC is complete. Once the ADC is complete the APND flag is automatically set to '1.' If interrupts are enabled (both AINTEN and G are set), when the APND flag is set to '1' an interrupt to the ACEx core is triggered. This bit can only be cleared by software but never set. If software writes a '1' to the APND bit, the register will simply retain its value. In order for a new conversion (both ADC and DAC) to begin the APND bit must be cleared.

The AEN bit is the ADDA block enable bit. If AEN is set to '1,' the converter will wait for a start condition to become true before a new conversion begins. The start condition can become true, only during the write of the ADCNTRL register, if the AEN bit is set and the APND flag is cleared. Once the start condition is true, a start pulse will be generated on the next rising edge of the system clock. The start condition will then begin the conversion (for both ADC and DAC).

Figure 12: ADCNTRL Register Bit Definition

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
APND	X	X	AINTEN	AEN	AMODE	ACHSEL[1:0]	

Table 10: ACHSEL Bit Definition

ACHSEL[1]	ACHSEL[0]	Channel (ADC) (AMODE = 0)	Channel (DAC) (AMODE = 1)	I/O Equiv.
0	0	ACH1	ACH1	Dedicated
0	1	ACH2	ACH2	G5
1	0	ACH3	ACH3	G6
1	1	ACH4 – ACH1	ACH4	G7

Note:

Once a conversion is in progress do not trigger another start condition (AEN set and APND cleared) otherwise the conversion will be killed with unpredictable results.

5.2 ADC Mode

The ADC mode is enabled when the AMODE bit is set to '0' while ACHSEL selects the analog input (ACH1 to ACH4). With the ADDA block enabled, the user software must reset the APND flag (even it if is already cleared) in order to trigger a conversion. Once the APND bit is reset, the ADC samples the analog input voltage and converts it to a 10-bit digital result. The result is then transferred to the 10-bit ADC register (ADDALO and ADDAHI) which then set the APND flag (see Figure 12(a)).

Relationship between 10-bit digital value and Analog Input:

$$V_{ADC} = \frac{V_{CH(x)} - A_{GND}}{A_{REF} - A_{GND}} * 1023$$

V_{ADC} is the 10-bit digital result of an ADV conversion.

$V_{CH(x)}$ is the Analog voltage applied to the selected input.

For example, suppose:

$V_{CH(x)} = 3V$
$A_{REF} = 5V$
$A_{GND} = 1V$

$$V_{ADC} = \frac{3 - 1}{5 - 1} * 1023 = 511 \text{ dec.}$$

In case $A_{GND} = 0$, the equation will be:

$$V_{ADC} = \frac{V_{CH(x)}}{A_{REF}} * 1023$$

In case the differential channel is selected, the relationship becomes:

$$V_{Diff} = \frac{(V_{CH4} - V_{CH1}) * A_{GAIN} - A_{GND}}{A_{REF} - A_{GND}} * 1023$$

Where A_{GAIN} is the differential stage gain, typically 18.

Recommendation in using the differential channel:

In order to maintain the specified differential gain and linearity, the differential amplifier network resistors R1, R2, R3 and R4 (see Figure 11a) are always connected to the respective differential inputs (ACH1 and ACH4). Care must be taken in designing the external circuit to consider the load seen at these inputs (typ. 95K) and the DC current flowing in the resistor network. This is especially important in order to respect the HALT currents where the current flowing in the resistors is not taken into consideration in the specified values. (See DC Electrical Characteristics for HALT current values.)

Note:

When the ADC is configured to convert the differential input (ACH4-ACH1), the first converted value has to be disregarded.

Software examples:

I. Subroutine to read an analog input (Polling Mode)

```
;***************************************************************************
;   Subroutine: ADC_READ
;   Accumulator holds the channel to be converted:
;           Acc = 0 -> ACH1
;           Acc = 1 -> ACH2
;           Acc = 2 -> ACH3
;           Acc = 3 -> (ACH4-ACH1) Differential
;   Accumulator holds not  valid area when returning from sub.
;   Reg. X points to the RAM area where to save the 10-bit converted value
;   Reg. X points to next Saving area word.
;***************************************************************************
;
;
ADC_READ:
    OR      A, #00001000b       ; Enable ADC and update channel
    ST      A, ADCNTRL          ; Write ADC control reg and start conversion
WaitEOC:
    IFBIT   APND, ADCNTRL       ; Test end of conversion
    JP      EndOfConv           ; Conversion ended, data available
    JP      WaitEOC             ; Conversion in progress
EndOfConv:
    LD      A, ADDALO           ; Load in Acc the 10-bit LSB
    ST      A, [X]              ; Store LSB in Saving area
    INC     X                   ; Point to Saving area MSB part
    LD      A, ADDAHI           ; Load in Acc the 10-bit MSB (2-bit)
    ST      A, [X]              ; Store MSB in Saving area
    INC     X
    RET
```

II. Program to read an analog input (Interrupt Mode)

The following example shows how to convert analog inputs using the ADC interrupt mode. The program performs a continuous scanning of the four analog inputs, the converted result is stored in saving location MemCH1 to MemCH4.

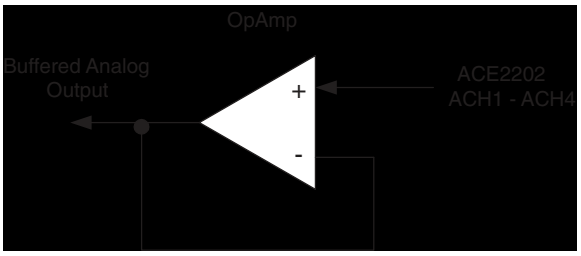
```
;
MemCH1_LO  equ 0              ; 10-bit conversion result for ACH1
MemCH2_LO  equ 2              ; 10-bit conversion result for ACH2
MemCH3_LO  equ 4              ; 10-bit conversion result for ACH3
MemCH4_LO  equ 6              ; 10-bit conversion result for ACH4-ACH1
SaveX      equ 8              ; RegX saving area in interrupt
SaveA      equ 9              ; Acc saving area in interrupt
ADC_Pntr   equ 0xA            ; Conversion buffer result pointer
;
;          ADC Configuration after Power On Reset
;
InitADC:
    LD     ADCNTRL, #00011000b; select ADC (CH1) in conversion mode
    LD     ADC_NTR, #MemCH1_LO; Init pointer to the CH1 saving area
    .
    .                         ; Rest of the Program
    .
;
;   ADC Interrupt service routine
;
ADC_INTR:
    LD     SaveX, XLO         ; save RegXlo in interrupt
    ST     A, SaveA           ; Save Accumulator
    LD     XLO, ADC_Pntr      ; RegX points to the Saving Area
    LD     A, ADDALO          ; Load Acc with 10-bit LSB result
    ST     A, [X]             ; Store LSB in saving area
    INC    X                  ; Point to saving area MSB
    LD     A, ADDAHI          ; Load Acc with 10-bit MSB result
    ST     A, [X]             ; Store MSB in saving area
    INC    X                  ; Point to next saving area channel
    IFGT   X, #MemCH4_LO      ; All four channels saved?
    LD     X, #MemCH1_LO      ; Yes -> Restart from CH1
    LD     ADC_Pntr,XLO       ; Update saving channel pointer
    LD     A, ADCNTRL         ; Load ADC Control current state
    AND    A, #00011011b      ; Mask APND -> to be cleared
    INC    A                  ; Point to next channel
    IFBIT  2, A               ; Last channel was ACH4?
    AND    A, #00011000b      ; Yes -> Restart from CH1
    ST     A, ADCNTRL         ; Update ADC Control register
    LD     XLO, SaveX         ; Restore regX
    LD     A, SaveA           ; Restore Acc
    RETI
;
;   Interrrupt Vector table
;
;
    ORG    0xFFE
    DW     ADC_INTR

    END
```

5.1 DAC Mode

The DAC mode is enabled when the AMODE bit is set to '1' while ACHSEL selects the analog output (ACH1 to ACH4). With the ADDA block enabled, the user software must write the lower and upper bytes of the DAC register (ADDALO, ADDAHI) then reset the APND flag (even if it is already cleared) in order to trigger a conversion. Once the APND bit is reset, the 10-bit DAC register will be transferred to Analog Voltage Generator (see Figure 11(b)) which performs the conversion.

Figure 13 shows a suggested circuit to interface with the selected analog output (ACH1 to ACH4) to an external analog circuit. This analog output buffer is necessary to de-couple the output from external excessive load and maintain the specified output level and linearity.

Figure 13. Analog Output Buffer



Relationship between the Analog output and 10-bit digital value:

$$A_{OUT} = \frac{V_{DAC} * (A_{REF} - A_{GND})}{1023} + A_{GND}$$

A_{OUT} is the analog voltage at the selected output in DAC mode.
V_{DAC} is the digital 10-bit word written to the DAC register.
A_{REF} is the Analog Reference.
A_{GND} is the Analog Ground.

For example, suppose:

V_{DAC} = 512 dec.
A_{REF} = 5V
A_{GND} = 1V

$$A_{OUT} = \frac{512 * (5 - 1)}{1023} + 1V = 3.002V$$

In case A_{GND} = 0, the equation will be:

$$A_{OUT} = \frac{V_{DAC} * A_{REF}}{1023}$$

Software example:

I. Program to generate a Saw-tooth waveform on ACH1

```
;
;   Continuously generating a Saw-Tooth waveform (Fout = 71 Hz @ 1MHz)
;
    LD      ADCNTRL, #00001100b; Enable DAC, use CH1
NewCycle:
    CLR     ADDALO              ; Clear 10-Bit LSB DAC register
    CLR     ADDAHI              ; Clear 10-Bit MSB DAC register
    RBIT    APND, ADCNTRL       ; ACH1 to AGND
SawLoop:
    INC     ADDALO              ; Increment DAC LSB register
    IFEQ    ADDALO, #00         ; Reached 255 ?
    INC     ADDAHI              ; Increment upper part
    RBIT    APND, ADCNTRL       ; xfer the 10-bit to the analog voltage gen.
    IFEQ    ADDAHI, #04         ; DAC Register higher than 1024?
    JP      NewCycle            ; Yes -> Start new cycle
    JP      SawLoop             ; Next step (every 14 cycles)
;
;
;
```

6.0 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), 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} and {T1RAHI, T1RALO}. 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 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 value) can either generate an interrupt and/or toggle the T1 output pin.

Timer 1 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.

6.1 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 Tables 11 and 12.

Table 11. TIMER1 Control Register (T1CNTRL)

T1CNTRL Register	Name	Function
Bit 7	T1C3	Timer TIMER1 control bit 3 (see Table 12)
Bit 6	T1C2	Timer TIMER1 control bit 2 (see Table 12)
Bit 5	T1C1	Timer TIMER1 control bit 1 (see Table 12)
Bit 4	T1C0	Timer TIMER1 run: 1 = Start timer, 0 = Stop timer; or Timer TIMER1 underflow interrupt pending flag in input capture mode
Bit 3	T1PND	Timer1 interrupt pending flag: 1 = Timer1 interrupt pending, 0 = Timer1 interrupt not pending
Bit 2	T1EN	Timer1 interrupt enable bit: 1 = Timer1 interrupt enabled, 0 = Timer1 interrupt disabled
Bit 1	M4S1	Capture type: 0 = Pulse capture, 1 = Cycle capture (see Table 12)
Bit 0	----------	Reserved

Table 12. TIMER1 Operating Modes

T1 C3	T1 C2	T1 C1	M4 S1	Timer Mode Source	Interrupt A	Timer Counts On
0	0	0	x	MODE 2	TIMER1 Underflow	T1 Pos. Edge
0	0	1	x	MODE 2	TIMER1 Underflow	T1 Neg. Edge
1	0	1	x	MODE 1 T1 Toggle	Autoreload T1RA	Instruction Clock
1	0	0	x	MODE 1 No T1 Toggle	Autoreload T1RA	Instruction Clock
0	1	0	x	MODE 3 Captures: T1 Pos. edge	Pos. T1 Edge	Instruction Clock
0	1	1	x	MODE 3 Captures: T1 Neg. Edge	Neg. T1 Edge	Instruction Clock
1	1	0	0	MODE 4 Difference Capture	Pos. to Neg.	Instruction Clock
1	1	0	1	MODE 4 Difference Capture	Pos. to Pos.	Instruction Clock
1	1	1	0	MODE 4 Difference Capture	Neg. to Pos.	Instruction Clock
1	1	1	1	MODE 4 Difference Capture	Neg. to Neg.	Instruction Clock

6.2 Mode 1. Pulse Width Modulation (PWM) Mode

In the PWM mode, the timer counts down at the instruction clock rate. When an underflow occurs, the timer register is reloaded from T1RA 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 load the T1RA register with an alternate PWM value. In addition, the timer can be configured to toggle the T1 output bit upon underflow. Configuring the timer to toggle T1 results in the generation of a signal outputted from port G2 with the width and duty cycle controlled by the values stored in the T1RA. A block diagram of the timer's PWM mode of operation is shown in Figure 14.

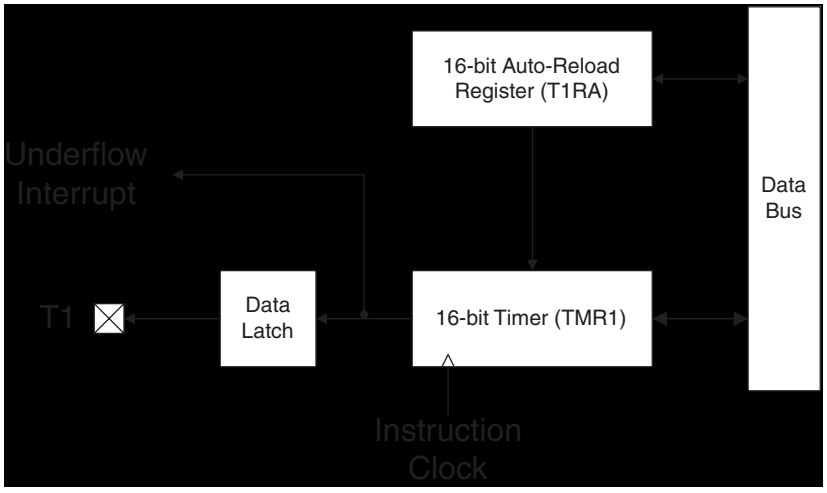
The timer has one interrupt (TMRI1) that is maskable through the T1EN bit of the T1CNTRL register. However, the core is only interrupted if the T1EN bit and the G (Global Interrupt enable) bit of the SR is set. If interrupts are enabled, the timer will generate an interrupt each time T1PND flags is set (whenever the 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 interrupt will be synchronous with every rising and falling edge of the T1 output signal. Generating interrupts only on rising or falling edges of T1 is achievable through appropriate handling of the T1EN bit or T1PND flag through software.

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 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 14. Pulse Width Modulation Mode



6.3 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 15.

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.

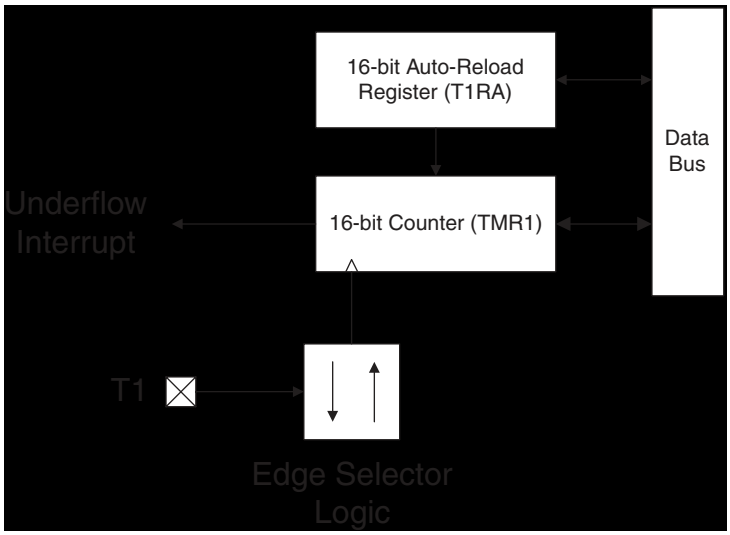
As with the PWM mode, when the counter underflows the counter is reloaded from the T1RA register and the count down proceeds from the loaded value. At every underflow, a pending flag (T1PND) located in the T1CNTRL register is set. Software must then clear the T1PND flag and can then load the T1RA register 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 (whenever 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, #00H
   - LD T1RALO, #0FFH
   - LD T1RAHI, #00H
5. Write the appropriate control value to the T1CNTRL register to select External Event Counter mode, to clock every falling edge, to set the enable bit, to clear the pending flag, and to start the counter. (See Table 11 and 12)
   - LD T1CNTRL, #34H (#00h) ; Setting the T1C0 bit starts the timer
6. When the counter underflows, the interrupt service routine must clear the T1PND flag and take whatever action is required once the number of events occurs. If the software wishes to merely count the number of events and the anticipated number may exceed 65,536, the interrupt service routine should record the number of underflows by incrementing a counter in memory. Software can then calculate the correct event count.
   - RBIT T1PND, T1CNTRL ; T1PND equals 3

Figure 15. External Event Counter Mode



6.4 Mode 3. Input Capture Mode

In the Input Capture mode, the timer is used to measure elapsed time between edges of an input signal. Once the timer is configured for this mode, the timer starts counting down immediately at the instruction clock rate. The Timer 1 will then transfer the current value of the TMR1 register into the T1RA register as soon as the selected edge of T1 is sensed. The input signal on T1 must have a pulse width equal to or greater than one instruction clock cycle. At every T1RA capture, software can then store the values into RAM to calculate the elapsed time between edges on T1. At any given time (with proper consideration of the state of T1) the timer can be configured to capture on positive-going or negative-going edges. A block diagram of the timer's Input Capture mode of operation is shown in Figure 16.

The timer 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. The Input Capture mode contains two interrupt pending flags 1) the TMR1 register capture in T1RA (T1PND) and 2) timer underflow (T1C0). If interrupts are enabled, the timer will generate an interrupt each time a pending flag is set (provided that the pending flag was previously cleared.) The interrupt service routine is responsible for proper handling of the T1PND flag, T1C0 flag, and the T1EN bit.

For this operating mode, the T1C0 control bit serves as the timer underflow interrupt pending flag. The Timer 1 interrupt service routine must read both the T1PND and T1C0 flags to determine the cause of the interrupt. A set T1C0 flag means that a timer underflow occurred whereas a set T1PND flag means that a capture occurred in T1RA. It is possible that both flags will be found set, meaning that both events occurred at the same time. The interrupt service routine should take this possibility into consideration.

Because the T1C0 bit is used as the underflow interrupt pending flag, it is not available for use as a start/stop bit as in the other modes.

The TMR1 register counts down continuously at the instruction clock rate starting from the time that the input capture mode is selected. (See Table 11 and 12) To stop the timer from running, you must change the mode to an alternate mode (PWM or External Event Counter) while resetting the T1C0 bit.

The input pins can be independently configured to sense positive-going or negative-going transitions. The edge sensitivity of pin T1 is controlled by bit T1C1 as indicated in Table 12.

The edge sensitivity of a pin can be changed without leaving the input capture mode even while the timer is running. This feature allows you to measure the width of a pulse received on an input pin.

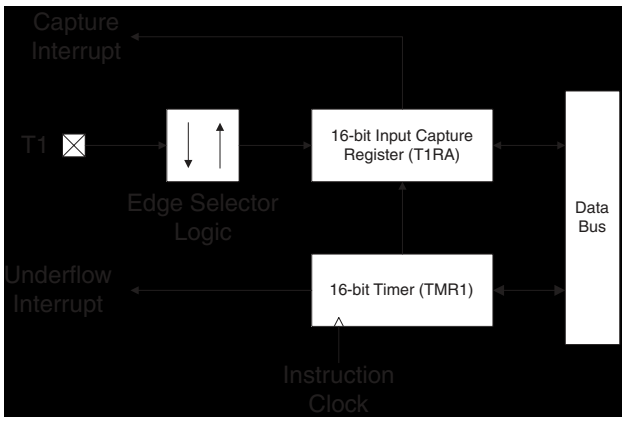
For example, the T1 pin can be programmed to be sensitive to a positive-going edge. When the positive edge is sensed, the TMR1 register contents is transferred to the T1RA register and a Timer 1 interrupt is generated. The Timer 1 interrupt service routine records the contents of the T1RA register, changes the edge sensitivity from positive to negative-going edge, and clears the T1PND flag. When the negative-going edge is sensed another Timer 1 interrupt is generated. The interrupt service routine reads the T1RA register again. The difference between the previous reading and the current reading reflects the elapsed time between the positive edge and negative edge of the T1 input signal i.e. the width of the positive-going pulse.

Remember that the Timer1 interrupt service routine must test the T1C0 and T1PND flags to determine the cause of the interrupt. If the T1C0 flag caused the interrupt, the interrupt service routine should record the occurrence of an underflow by incrementing a counter in memory or by some other means. The software that calculates the elapsed time between captures should take into account the number of underflow that occurred when making its calculation.

The following steps show how to properly configure Timer 1 to operate in the Input Capture mode.

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. With the timer stopped, load the initial time into the TMR1 register (typically the value is 0xFFFF.)
   - LD TMR1LO, #0FFH
   - LD TMR1HI, #00H
5. Write the appropriate control value to the T1CNTRL register to select Input Capture mode, to sense the appropriate edge, to set the enable bit, and to clear the pending flags. (See Table 11 and 12)
   - LD T1CNTRL, #64H ; T1C1 is the edge select bit
6. 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 16. Input Capture Mode



6.5 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.

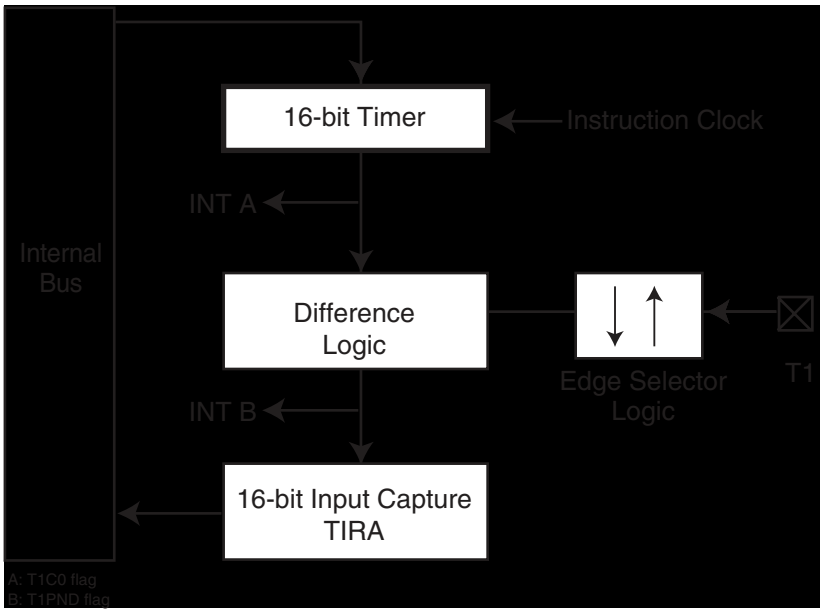
The Difference Capture mode eliminates the need for software intervention and allows for capturing very short pulse or cycle widths. it can be configured to capture the elapsed time between:

1. positive to negative-going edges
2. positive to positive-going edges
3. negative to positive-going edges
4. negative to negative-going edges

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 17. Difference Capture Mode



7.0 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 (T0PND) bit that is set every 8192 cycles (initially 4096 cycles after a reset). The T0PND flag is set every other time the timer overflows (transitions from 0xFFF 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 T0PND bit or vector to an interrupt subroutine. In order to interrupt on a T0PND, software must be sure to enable the Timer 0 interrupt enable (T0INTEN) 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 T0PND bit before exiting the routine. Timer 0 supports the following functions:

1. Start up delay from HALT mode
2. Watchdog pre-scaler (See Section 8.0 for details.)

The T0INTEN 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 T0INTEN bit is reset to 0.

The T0PND 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 9 for details.

7.0 Watchdog timer

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-scaler 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 19). 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.

Figure 18. Timer 0 Control Register (T0CNTRL)

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
WKINTEN	x	x	x	x	x	T0PND	T0INTEN

Figure 19. Watchdog Server 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

9.0 Multi-Input Wakeup/Interrupt Block

The Multi-Input Wakeup (MIW)/Interrupt contains three memory-mapped registers associated with this circuit: WKEDG (Wakeup Edge), WKEN (Wakeup Enable), and WKPND (Wakeup Pending). Each register has 8-bits with each bit corresponding to an input pins as shown in Figure 20. All three registers are initialized to zero upon reset.

The WKEDG register establishes the edge sensitivity for each of the wake-up input pin: either positive going-edge (0) or negative-going edge (1).

The WKEN register enables (1) or disables (0) each of the port pins for the Wakeup/Interrupt function. The wakeup I/Os used for the Wakeup/Interrupt function must also be configured as an input pin in its associated port configuration register. However, an interrupt of the core will not occur unless interrupts are enabled for the block via bit 7 of the T0CNTRL register (see Figure 18) and the G (global interrupt enable) bit of the SR is set.

The WKPND register contains the pending flags corresponding to each of the port pins (1 for wakeup/interrupt pending, 0 for wakeup/interrupt not pending).

To use the Multi-Input Wakeup/Interrupt circuit, perform the steps listed below. Performing the steps in the order shown will prevent false triggering of a Wakeup/Interrupt condition. This same procedure should be used following any type of reset because the wakeup inputs are left floating after resets resulting in unknown data on the port inputs.

1. Clear the WKEN register.
   - CLR WKEN
2. If necessary, write to the port configuration register to select the desired port pins to be configured as inputs.
   - RBIT 4, PORTGC ; G4
3. If necessary, write to the port data register to select the desired port pins input state.
   - SBIT 4, PORTGD ; Pull-up
4. Write the WKEDG register to select the desired type of edge sensitivity for each of the pins used.
   - LD WKEDG, #0FFH ; All negative-going edges
5. Clear the WKPND register to cancel any pending bits.
   - CLR WKPND
6. Set the WKEN bits associated with the pins to be used, thus enabling those pins for the Wakeup/Interrupt function.
   - LD WKEN, #10H ; Enabling G4

Once the Multi-Input Wakeup/Interrupt function has been configured, a transition sensed on any of the I/O pins will set the corresponding bit in the WKPND register. The WKPND bits, where the corresponding enable (WKEN) bits are set, will bring the device out of the HALT mode and can also trigger an interrupt if interrupts are enabled. The interrupt service routine can read the WKPND register to determine which pin sensed the interrupt.

The interrupt service routine or other software should clear the pending bit. The device will not enter HALT mode as long as a WKPND pending bit is pending and enabled. The user has the responsibility of clearing the pending flags before attempting to enter the HALT mode.

Upon reset, the WKEDG register is configured to select positive-going edge sensitivity for all wakeup inputs. If the user wishes to change the edge sensitivity of a port pin, use the following procedure to avoid false triggering of a Wakeup/Interrupt condition.

1. Clear the WKEN bit associated with the pin to disable that pin.
2. Write the WKEDG register to select the new type of edge sensitivity for the pin.
3. Clear the WKPND bit associated with the pin.
4. Set the WKEN bit associated with the pin to re-enable it.

PORTG provides the user with three fully selectable, edge sensitive interrupts that are all vectored into the same service subroutine. The interrupt from PORTG shares logic with the wakeup circuitry. The WKEN register allows interrupts from PORTG to be individually enabled or disabled. The WKEDG register specifies the trigger condition to be either a positive or a negative edge. The WKPND register latches in the pending trigger conditions.

Since PORTG is also used for exiting the device from the HALT mode, the user can elect to exit the HALT mode either with or without the interrupt enabled. If the user elects to disable the interrupt, then the device restarts execution from the point at which it was stopped (first instruction cycle of the instruction following HALT mode entrance instruction). In the other case, the device finishes the instruction that was being executed when the part was stopped and then branches to the interrupt service routine. The device then reverts to normal operation.

When more than one MIW pin is enabled in the user program must avoid clearing the respective WKPND bit with the instruction RBIT to inadvertently clear another MIW becoming pending within the RBIT instruction execution. The software should use instead a LD WKPND,#CONF instruction where CONF corresponds to a mask clearing the MIW bit in question but setting to "1" the remaining ones. For example, if the port G0 and G1 are used and enabled as MIW inputs the clearing procedure should be:

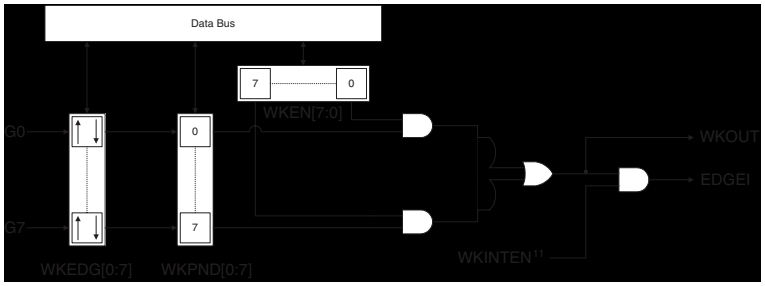
```
MIW_Handler:
IFBIT0,WKPND              ; MIW active on port G0?
JP   _MIW_G0              ; Yes, service MIW-0
IFBIT1,WKPND              ; MIW active on port G1?
JP   _MIW_G1              ; Yes, service MIW-1
...                       ; No MIW active continue the loop
_MIW_G0:
LD   WKPND,#11111110b     ; Clear pending bit0
...                       ; Continue the loop
_MIW_G1:
LD   WKPND,#11111101b     ; Clear pending bit1
...                       ; Continue the loop
```

For additional information see the "ACEx Device Help" guide available in the ACEx development tool suite.

Figure 20. Multi-input Wakeup (MIW) Register bit assignments

WKEDG, WKEN, WKPND							
Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
G7	G6	G5	G4	G3	G2	G1	G0

Figure 21. Multi-input Wakeup (MIW) Block Diagram



[11] WKINTEN: Bit 7 of T0CNTRL

10.0 I/O Port

The eight I/O pins are bi-directional (see Figure 22) with the exception of G3 which is always an input with weak pull-up. 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.

10.1 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-impedence input. Table 14 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.

Figure 22. PORTG Logic Diagram

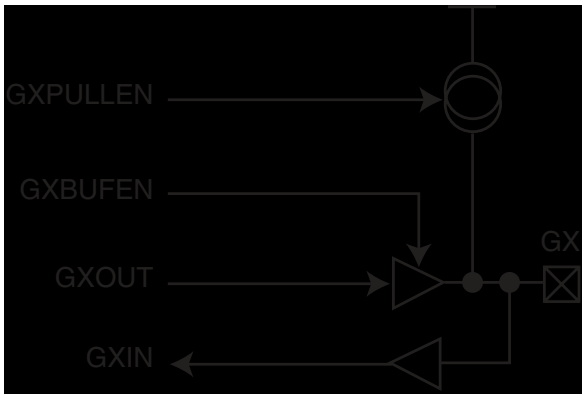


Figure 23. I/O Register bit assignments (PORTGC, PORTGD, PORTGD)

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
G7	G6	G5	G4	G3[12]	G2	G1	G0

[12]G3 is always an input with weak pull-up

Table 13. I/O configuration options

Configuration Bit	Data Bit	Port Pin Configuration
0	0	High-impedence 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

11.0 In-circuit Programming Specification[13,14]

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

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.

The serial data timing for the four-wire interface is shown in Figure 25 and the programming protocol is shown in Figure 24.

11.1 Write Sequence

The external programmer brings the ACEx microcontroller into programming mode by applying a super voltage level to the LOAD pin. The external programmer then needs to set the LOAD pin to 5V 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} (500ns) 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 24 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 24 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 5V to initiate a normal command cycle.

11.2 Read Sequence

When reading the device after a write, the external programmer must set the LOAD signal to 5V 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 24 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

[13]For further information see Application Note AN-2001.

[14]During in-circuit programming, G5 (pin 6) must be either not connected or driven high.

Figure 24. Programming Protocol[14]

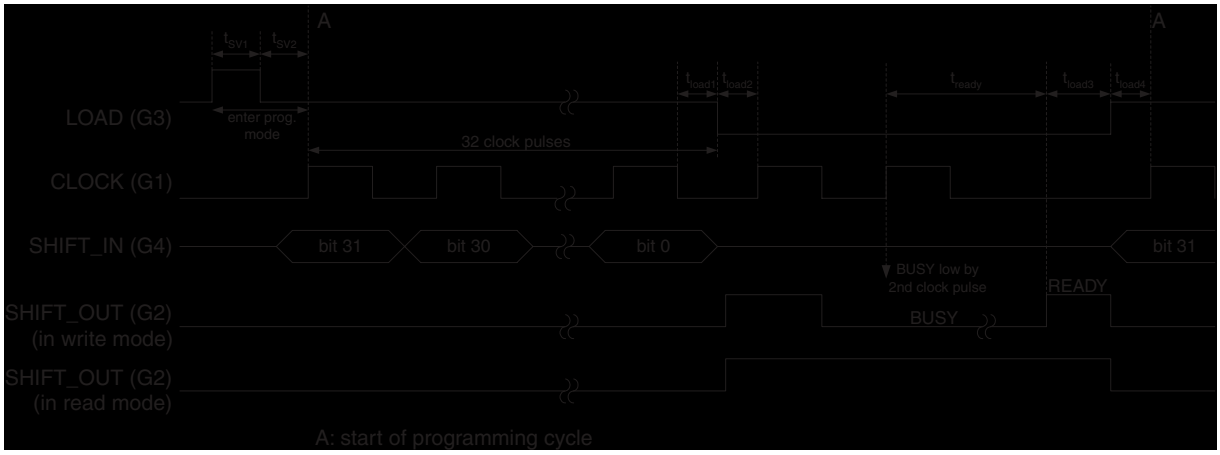
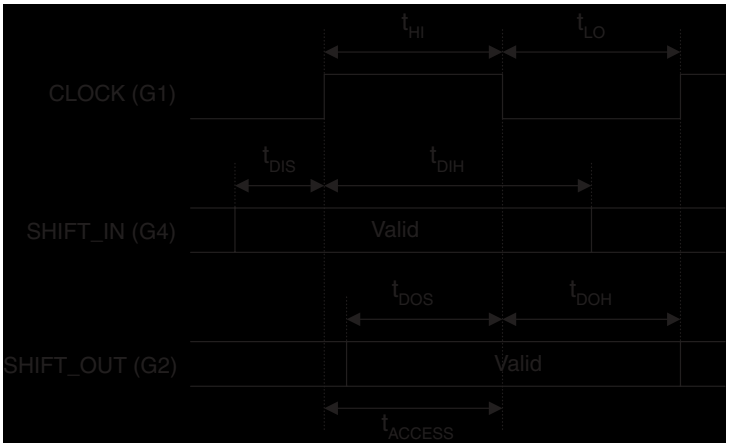


Figure 25. Serial Data Timing



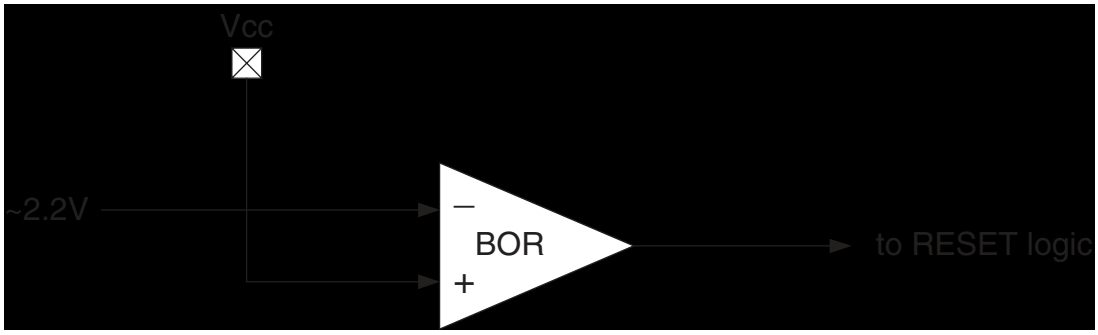
12.0 Brown Out Reset

The Brown-out Reset (BOR) function is used to hold the device in reset when V_{CC} drops below a fixed threshold. (See BOR Electrical Characteristics for threshold voltage.) While in reset, the device is held in its initial condition until V_{CC} rises above the threshold value. Shortly after V_{CC} 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.

On the devices, the BOR should be used in situations when V_{CC} rises and falls slowly and in situations when V_{CC} 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 when V_{CC} does not fall below ~1.5V. The Power-on Reset circuit works best when V_{CC} starts from zero and rises sharply. So in applications where V_{CC} 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 26. BOR Circuit Block Diagram



13.0 RESET block

When a RESET sequence is initiated, all I/O registers will be reset setting all I/Os to high-impedence inputs. The system clock is restarted after the required clock start-up delay. A reset is generated by any one of the following three conditions:

- Power-on Reset (as described in Section 14.0)
- Brown-out Reset (as described in Section 12.0)
- Watchdog Reset (as described in Section 8.0)
- External Reset18 (as described in Section 14.0)

14.0 Power-On-Reset

The Power-On Reset (POR) circuit is guaranteed to work if the rate of rise of V_{CC} is no slower than 10ms/1volt. The POR circuit was designed to respond to fast low to high transitions between 0V and V_{CC}. The circuit will not work if V_{CC} does not drop to 0V before the next power-up sequence. In applications where 1) the V_{CC} rise is slower than 10ms/1 volt or 2) V_{CC} does not drop to 0v before the next power-up sequence the external reset option should be used.

The external reset provides a way to properly reset the ACEx microcontroller if POR cannot be used in the application. The external reset pin contains an internal pull-up resistor. Therefore, to reset the device the reset pin should be held low for at least 2ms so that the internal clock has enough time to stabilize.

15.0 CLOCK

The ACEx microcontroller has an on-board oscillator trimmed to a frequency of 2MHz who is divided down by two yielding a 1MHz frequency. (See AC Electrical Characteristics.) Upon power-up, the on-chip oscillator runs continuously unless entering HALT mode or using an external clock source.

If required, an external oscillator circuit may be used depending on the states of the CMODE bits of the initialization register. (See Table 15) When the device is driven using an external clock, the clock input to the device (G1/CKI) can range between DC to 4MHz. For external crystal configuration, the output clock (CKO) is on the G0 pin. (See Figure 28) If an external crystal or RC is used, internally the input frequency (CKI) is divided-down by four to yield the corresponding instruction clock. If the device is configured for an external square clock, it will not be divided.

Table 15. CMODEx Bit Definition

CMODE[1]	CMODE[0]	Clock Type
0	0	Internal 1 MHz clock
0	1	External square clock
1	0	External crystal/resonator
1	1	External RC clock

Figure 27. BOR and POR Circuit Relationship Diagram

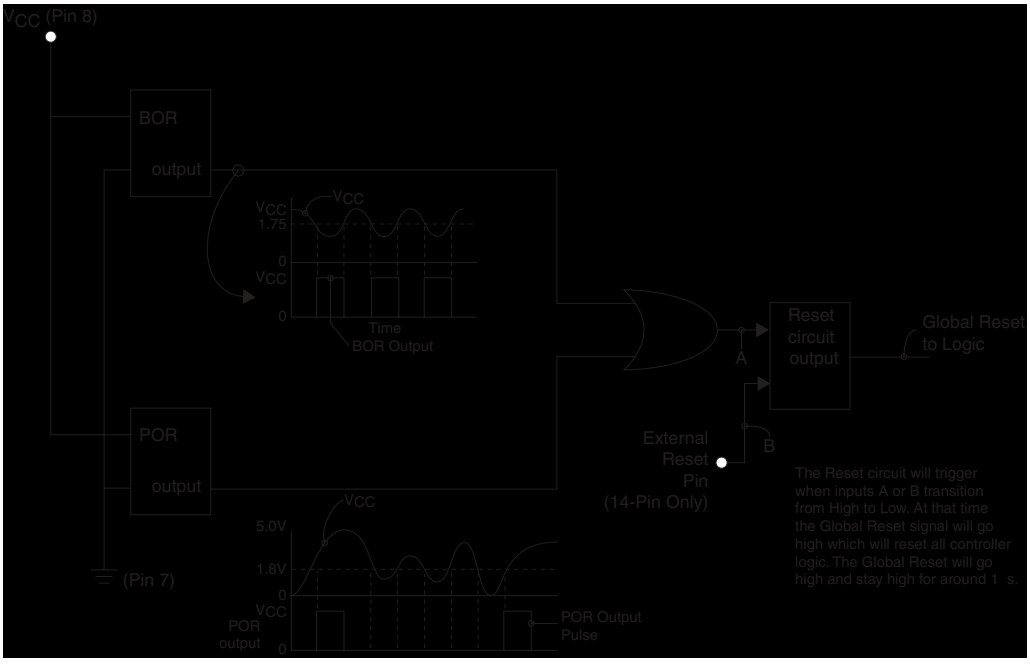
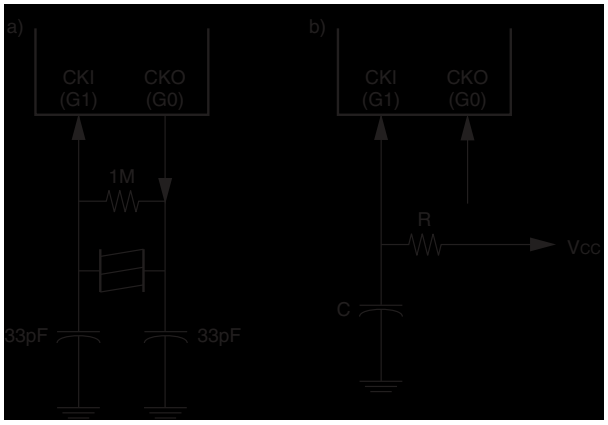


Figure 28. Crystal (a) and RC (b) Oscillator Diagrams



16.0 HALT Mode

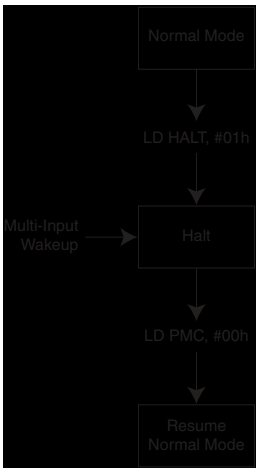
The HALT mode is a power saving feature that almost completely shuts down the device for current conservation. The device is placed into HALT mode by setting the HALT enable bit (EHALT) of the HALT register through software using only the "LD M, #" instruction. EHALT is a write only bit and is automatically cleared upon exiting HALT. When entering HALT, the internal oscillator and all the on-chip systems including the LBD and the BOR circuits are shut down.

The device can exit HALT mode only by the MIW circuit. Therefore, prior to entering HALT mode, software must configure the MIW circuit accordingly. (See Section 9) After a wakeup from HALT, a 1ms start-up delay is initiated to allow the internal oscillator to stabilize before normal execution resumes. Immediately after exiting HALT, software must clear the Power Mode Clear (PMC) register by only using the "LD M, #" instruction. (See Figure 30)

Figure 29. HALT Register Definition

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

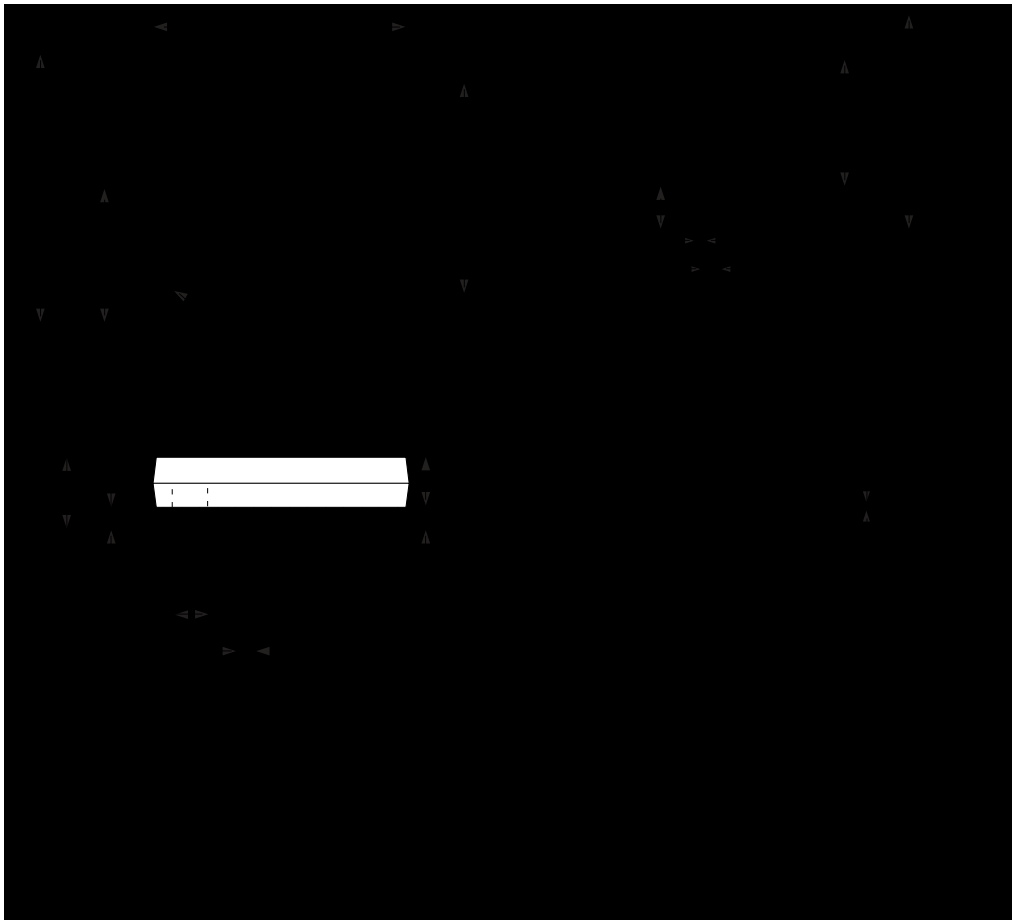
Figure 30. Recommended HALT Flow



Ordering Information

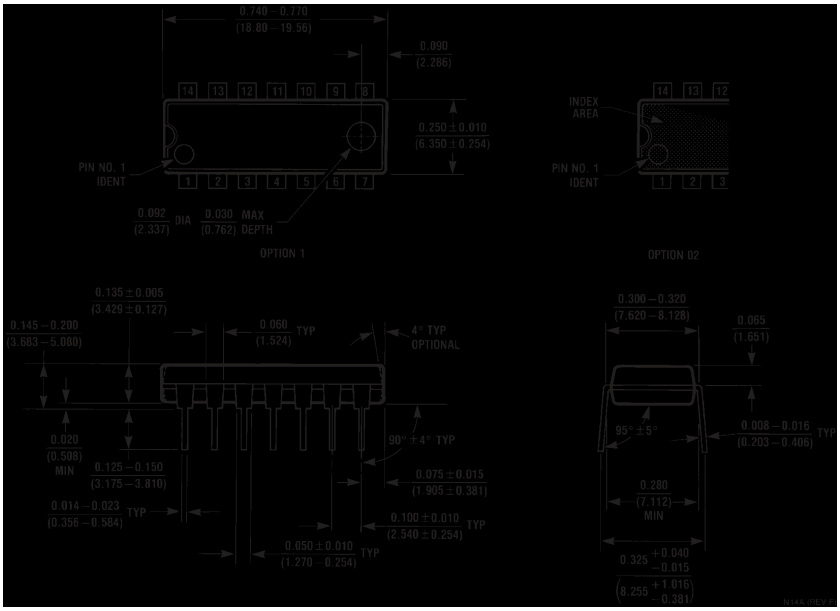
Part Number	Core Type			Max. # I/Os	Max. # Analog I/Os	Program Memory Size		Operating Voltage Range	Temperature Range			Package		Tape & Reel
	0	1	2	8	3 (mux.), 1 (dedicated)	1K	2K	2.4 – 5.5V	0 to 70°C	-40 to +85C	-40 to +125°C	14-pin TSSOP	14-pin DIP	
ACE2202MT14			X	X	X		X	X	X			X		
ACE2202MT14X			X	X	X		X	X	X			X		X
ACE2202N14			X	X	X		X	X	X				X	

Physical Dimensions inches (millimeters) unless otherwise noted



TSSOP Package (MT14)
Order Number ACE2202MT14
Package Number MTC014

Physical Dimensions inches (millimeters) unless otherwise noted



14-Pin DIP (N14)
Order Number ACE2202N14
Package Number N14A

ACEx Development Tools

General Information

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

ACEx Starter Kit includes:

Programmer Board
Simulator Software
Programmer Software
Assembler and Manuals
Cables and samples devices
DIP programming sockets

Programmer board: Interfaces with a PC through a Windows program using the serial communication port. This board is intended for engineering prototype and can be used for small volume production. 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.

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 offer two solutions for the simplification of the breadboard operation so that ACEx Applications can be quickly tested.

1) ACEDEMO is can be used for general purpose applications
2) ACETXRX 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.

Ordering P/Ns

Starter Kit: ACESTART1101
ACESTART1202

Programming Adapters:
DIP8 - ACESDIP8
DIP14 - ACESDIP14
TSSOP8 - ACESTSSOP8
SO8 - ACESSOP8
SO14 - ACESSOP15

Emulator Kit: ACEICE2 (110Vac)
ACEICE2EU (220Vac)

Prototype Boards:
ACEDEMO
ACETXRX (specify RF freq. 433 or 315MHz)

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
Fax: +44 (0) 1793-856858
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