Xinqun

Cost-Effective A/D OTP MCU

BX66R004

Revision: V1.00 Date: February 24, 2025

Table of Contents

Features

CPU Features

- Operating voltage
  - f_{SYS}=8MHz: 2.0V~5.5V
- Up to 0.5μs instruction cycle with 8MHz system clock at V_{DD}=5V
- Power down and wake-up functions to reduce power consumption
- Oscillator types
  - Internal High Speed 8MHz RC – HIRC
  - Internal Low Speed 32kHz RC – LIRC
- Multi-mode operation: FAST, SLOW, IDLE and SLEEP
- Fully integrated internal oscillators require no external components
- All instructions executed in one or two instruction cycles
- Table read instructions
- 61 powerful instructions
- 4-level subroutine nesting
- Bit manipulation instruction

Peripheral Features

- OTP Program Memory: (2K-16)×15
- Data Memory: 96×8
- OTP ROM Parameter Program function – ORPP
- Watchdog Timer function
- Up to 18 bidirectional I/O lines
- Two external interrupt lines shared with I/O pins
- Single 8-bit programmable Timer/Event Counter with PWM output
  - 10-channel PWM function with three kinds of independently adjustable duty cycle
- Single 16-bit programmable Timer/Event Counter
- 10 external channel 8-bit resolution A/D converter
- Single Time-Base function for generation of fixed time interrupt signals
- Low voltage reset function
- Software controlled 7-SCOM lines LCD driver with 1/2 bias
- Package types: 16-pin NSOP, 20-pin NSOP/SOP/SSOP

General Description

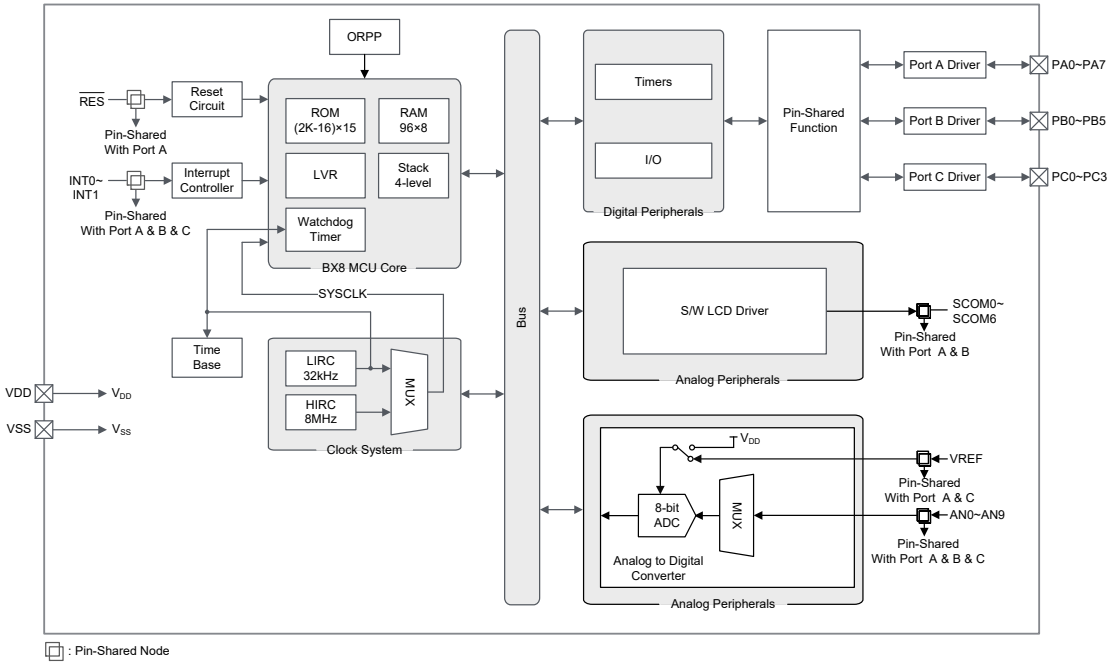
This device is an OTP type 8-bit high performance RISC architecture microcontroller, designed for cost-effective product applications.

Analog features include a multi-channel 8-bit A/D converter function. Two extremely flexible Timer/Event Counters providing functions for timing, event counting and pulse width measurement functions. Protective features such as an internal Watchdog Timer and Low Voltage Reset coupled with excellent noise immunity and ESD protection ensure that reliable operation is maintained in hostile electrical environments.

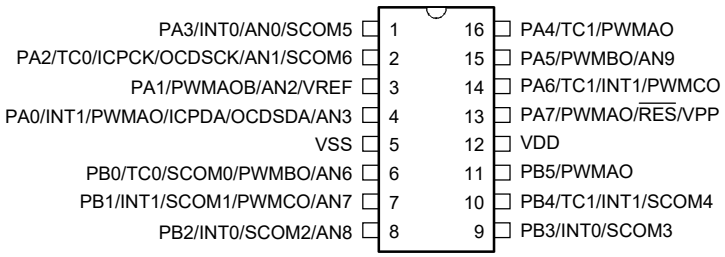
A full choice of internal high and low speed oscillators are provided and the two fully integrated system oscillators require no external components for their implementation. The ability to operate and switch dynamically between a range of operating modes using different clock sources gives users the ability to optimize microcontroller operation and minimize power consumption.

The inclusion of flexible I/O programming features, Time-Base function along with many other features ensure that the device will find excellent use in applications such as electronic metering, environmental monitoring, handheld instruments, household appliances, electronically controlled tools, motor driving in addition to many others.

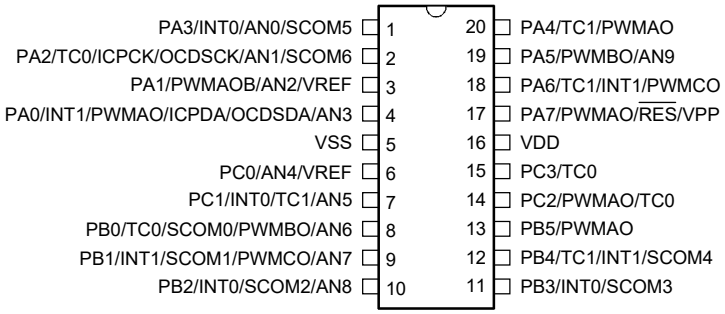
Block Diagram



Pin Assignment



BX66R004/BX66V004
16 NSOP-A



BX66R004/BX66V004
20 NSOP-A/SOP-A/SSOP-A

Note: 1. If the pin-shared pin functions have multiple outputs, the desired pin-shared function is determined by the corresponding software control bits.

2. The OCDSDA and OCDSCK pins are supplied for the OCDS dedicated pins and only available for the BX66V004 device (Flash type) which is the OCDS EV chip for the BX66R004 device (OTP type).

3. For less pin-count package types there will be unbonded pins which should be properly configured to avoid unwanted current consumption resulting from floating input conditions. Refer to the "Standby Current Considerations" and "Input/Output Ports" sections.

4. The VPP pin is the High Voltage input for OTP programming and only available for the BX66R004 device.

Pin Description

The function of each pin is listed in the following table, however the details behind how each pin is configured is contained in other sections of the datasheet. As the Pin Description table shows the situation for the package with the most pins, not all pins in the tables will be available on smaller package sizes.

Pin Name	Function	OPT	I/T	O/T	Description
PA0/INT1/PWMAO/AN3/ ICPDA/OCDSDA	PA0	PAPU PAWU PAS0	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up.
	INT1	INTEG INTC1 PAS0 IFS	ST	—	External interrupt input 1
	PWMAO	PAS0	—	CMOS	PWMA signal output
	AN3	PAS0	AN	—	A/D Converter external input channel
	ICPDA	—	ST	CMOS	ICP Data/Address pin
	OCDSDA	—	ST	CMOS	OCDS Address/Data pin, for EV chip only

Pin Name	Function	OPT	I/T	O/T	Description
PA1/PWMAOB/AN2/VREF	PA1	PAPU PAWU PAS0	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up.
	PWMAOB	PAS0	—	CMOS	PWMA signal inverter output
	AN2	PAS0	AN	—	A/D Converter external input channel
	VREF	PAS0	AN	—	A/D Converter external reference voltage input
PA2/TC0/AN1/SCOM6/ ICPCK/OCDSCK	PA2	PAPU PAWU PAS0	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up.
	TC0	PAS0 IFS	ST	—	Timer/Event Counter 0 clock input
	AN1	PAS0	AN	—	A/D Converter external input channel
	SCOM6	PAS0	—	AN	Software LCD COM output
	ICPCK	—	ST	—	ICP clock pin
	OCDSCK	—	ST	—	OCDS clock pin, for EV chip only
PA3/INT0/AN0/SCOM5	PA3	PAPU PAWU PAS0	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up.
	INT0	INTEG INTC0 PAS0 IFS	ST	—	External interrupt input 0
	AN0	PAS0	AN	—	A/D Converter external input channel
	SCOM5	PAS0	—	AN	Software LCD COM output
PA4/TC1/PWMAO	PA4	PAPU PAWU PAS1	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up.
	TC1	PAS1 IFS	ST	—	Timer/Event Counter 1 clock input
	PWMAO	PAS1	—	CMOS	PWMA signal output
PA5/PWMBO/AN9	PA5	PAPU PAWU PAS1	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up.
	PWMBO	PAS1	—	CMOS	PWMB signal output
	AN9	PAS1	AN	—	A/D Converter external input channel
PA6/TC1/INT1/PWMCO	PA6	PAPU PAWU PAS1	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up.
	TC1	PAS1 IFS	ST	—	Timer/Event Counter 1 clock input
	INT1	INTEG INTC1 PAS1 IFS	ST	—	External interrupt input 1
	PWMCO	PAS1	—	CMOS	PWMC signal output
PA7/PWMAO/$\overline{\text{RES}}$/VPP	PA7	PAWU PAS1	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up. This I/O is pin-shared with VPP and could result in increased current consumption if it is set to output high.
	PWMAO	PAS1	—	CMOS	PWMA signal output
	$\overline{\text{RES}}$	PAS1	ST	—	External reset input
	VPP	PAS1	PWR	—	High Voltage input for OTP programming, not available for EV chip

Pin Name	Function	OPT	I/T	O/T	Description
PB0/TC0/SCOM0/ PWMBO/AN6	PB0	PBPU PBS0	ST	CMOS	General purpose I/O. Register enabled pull-up
	TC0	PBS0 IFS	ST	—	Timer/Event Counter 0 clock input
	SCOM0	PBS0	—	AN	Software LCD COM output
	PWMBO	PBS0	—	CMOS	PWMB signal output
	AN6	PBS0	AN	—	A/D Converter external input channel
PB1/INT1/SCOM1/ PWMCO/AN7	PB1	PBPU PBS0	ST	CMOS	General purpose I/O. Register enabled pull-up
	INT1	INTEG INTC1 PBS0 IFS	ST	—	External interrupt input 1
	SCOM1	PBS0	—	AN	Software LCD COM output
	PWMCO	PBS0	—	CMOS	PWMC signal output
	AN7	PBS0	AN	—	A/D Converter external input channel
PB2/INT0/SCOM2/AN8	PB2	PBPU PBS0	ST	CMOS	General purpose I/O. Register enabled pull-up
	INT0	INTEG INTC0 PBS0 IFS	ST	—	External interrupt input 0
	SCOM2	PBS0	—	AN	Software LCD COM output
	AN8	PBS0	AN	—	A/D Converter external input channel
PB3/INT0/SCOM3	PB3	PBPU PBS0	ST	CMOS	General purpose I/O. Register enabled pull-up
	INT0	INTEG INTC0 PBS0 IFS	ST	—	External interrupt input 0
	SCOM3	PBS0	—	AN	Software LCD COM output
PB4/TC1/INT1/SCOM4	PB4	PBPU PBS1	ST	CMOS	General purpose I/O. Register enabled pull-up
	TC1	PBS1 IFS	ST	—	Timer/Event Counter 1 clock input
	INT1	INTEG INTC1 PBS1 IFS	ST	—	External interrupt input 1
	SCOM4	PBS1	—	AN	Software LCD COM output
PB5/PWMAO	PB5	PBPU PBS1	ST	CMOS	General purpose I/O. Register enabled pull-up
	PWMAO	PBS1	—	CMOS	PWMA signal output
PC0/AN4/VREF	PC0	PCPU PCS0	ST	CMOS	General purpose I/O. Register enabled pull-up
	AN4	PCS0	AN	—	A/D Converter external input channel
	VREF	PCS0	AN	—	A/D Converter external reference voltage input
PC1/INT0/TC1/AN5	PC1	PCPU	ST	CMOS	General purpose I/O. Register enabled pull-up
	INT0	INTEG INTC0 IFS	ST	—	External interrupt input 0
	TC1	IFS	ST	—	Timer/Event Counter 1 clock input
	AN5	PCS0	AN	—	A/D Converter external input channel

Pin Name	Function	OPT	I/T	O/T	Description
PC2/TC0/PWMAO	PC2	PCPU PCS0	ST	CMOS	General purpose I/O. Register enabled pull-up
	TC0	PCS0 IFS	ST	—	Timer/Event Counter 0 clock input
	PWMAO	PCS0	—	CMOS	PWMA signal output
PC3/TC0	PC3	PCPU	ST	CMOS	General purpose I/O. Register enabled pull-up
	TC0	IFS	ST	—	Timer/Event Counter 0 clock input
VDD	VDD	—	PWR	—	Positive power supply
VSS	VSS	—	PWR	—	Negative power supply, ground

Legend: I/T: Input type; O/T: Output type;
OPT: Optional by register option; PWR: Power;
ST: Schmitt Trigger input; CMOS: CMOS output;
NMOS: NMOS output; AN: Analog signal.

Absolute Maximum Ratings

Supply Voltage ..... V_{SS}-0.3V to 6.0V

Input Voltage ..... V_{SS}-0.3V to V_{DD}+0.3V

Storage Temperature ..... -60°C to 150°C

Operating Temperature ..... -40°C to 85°C

I_{OH} Total ..... -80mA

I_{OL} Total 80mA

Total Power Dissipation 500mW

Note: These are stress ratings only. Stresses exceeding the range specified under "Absolute Maximum Ratings" may cause substantial damage to the device. Functional operation of the device at other conditions beyond those listed in the specification is not implied and prolonged exposure to extreme conditions may affect device reliability.

D.C. Characteristics

For data in the following tables, note that factors such as oscillator type, operating voltage, operating frequency, pin load conditions, temperature and program instruction type, etc., can all exert an influence on the measured values.

Operating Voltage Characteristics

Ta=-40°C~85°C

Symbol	Parameter	Test Conditions	Min.	Typ.	Max.	Unit
V_{DD}	Operating Voltage – HIRC	$f_{SYS}=f_{HIRC}$=8MHz	2.0	—	5.5	V
	Operating Voltage – LIRC	$f_{SYS}=f_{LIRC}$=32kHz	2.0	—	5.5	V

Standby Current Characteristics

Ta=25°C, unless otherwise specified

Symbol	Standby Mode	Test Conditions		Min.	Typ.	Max.	Max. @85°C	Unit
		V_{DD}	Conditions					
I_{STB}	SLEEP Mode	2V	WDT off	—	0.45	0.80	7.00	μA
		3V		—	0.45	0.90	8.00	
		5V		—	0.5	2.0	10.0	
		2V	WDT on	—	1.5	3.0	5.5	μA
		3V		—	1.8	3.6	6.5	
		5V		—	3	5	10	
	IDLE0 Mode – LIRC	2V	f_{SUB} on	—	2.4	4.0	8.0	μA
		3V		—	3	5	9	
		5V		—	5	10	11	
	IDLE1 Mode – HIRC	2V	f_{SUB} on, f_{SYS}=8MHz	—	288	400	480	μA
		3V		—	360	500	600	
		5V		—	600	800	960	

Note: When using the characteristic table data, the following notes should be taken into consideration:

1. Any digital inputs are set in a non-floating condition and the I/O pin-shared with VPP is not setup in an output high condition.
2. All measurements are taken under conditions of no load and with all peripherals in an off state.
3. There are no DC current paths.
4. All Standby Current values are taken after a HALT instruction execution thus stopping all instruction execution.

Operating Current Characteristics

Ta=-40°C~85°C

Symbol	Operating Mode	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
I_{DD}	SLOW Mode – LIRC	2V	f_{SYS}=32kHz	—	12	24	μA
		3V		—	15	30	
		5V		—	30	50	
	FAST Mode – HIRC	2V	f_{SYS}=8MHz	—	0.6	1.0	mA
		3V		—	0.8	1.2	
		5V		—	1.6	2.4	

Note: When using the characteristic table data, the following notes should be taken into consideration:

1. Any digital inputs are set in a non-floating condition and the I/O pin-shared with VPP is not setup in an output high condition.
2. All measurements are taken under conditions of no load and with all peripherals in an off state.
3. There are no DC current paths.
4. All Operating Current values are measured using a continuous NOP instruction program loop.

A.C. Characteristics

For data in the following tables, note that factors such as oscillator type, operating voltage, operating frequency and temperature, etc., can all exert an influence on the measured values.

High Speed Internal Oscillator – HIRC – Frequency Accuracy

During the program writing operation the writer will trim the HIRC oscillator at a user selected HIRC frequency and user selected voltage of either 3V or 5V.

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Temp.				
f_{HIRC}	8MHz Writer Trimmed HIRC Frequency	3V/5V	25°C	-1.5%	8	+1.5%	MHz
			-40°C~85°C	-4%	8	+4%	
		2.0V~5.5V	25°C	-3%	8	+3%	
			-40°C~85°C	-5%	8	+5%	

Note: 1. The 3V/5V values for V_{DD} are provided as these are the two selectable fixed voltages at which the HIRC frequency is trimmed by the writer.

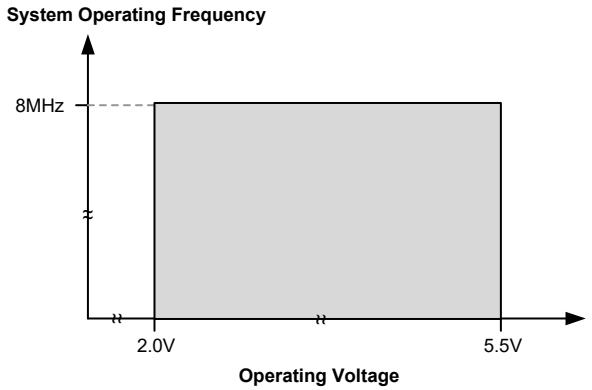
2. The row below the 3V/5V trim voltage row is provided to show the values for the full V_{DD} range operating voltage. It is recommended that the trim voltage is fixed at 3V for application voltage ranges from 2.0V to 3.6V and fixed at 5V for application voltage ranges from 3.3V to 5.5V.
3. The minimum and maximum tolerance values provided in the table are for the frequency at which the writer trims the HIRC oscillator.
4. In actual applications, it is important to note that a 0.1μF decoupling capacitor should be connected between VDD and VSS and located as close to the VDD pin as possible to avoid the frequency tolerance exceeding the specification.

Low Speed Internal Oscillator – LIRC – Frequency Accuracy

Ta=-40°C ~85°C, unless otherwise specified

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Temp.				
f_{LIRC}	LIRC Frequency	2.0V~5.5V	25°C	-20%	32	+20%	kHz
			-40°C~85°C	-50%	32	+60%	
t_{START}	LIRC Start Up Time	—	—	—	—	500	μs

Operating Frequency Characteristic Curves



System Start Up Time Characteristics

Ta=-40°C~85°C

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
t_{SST}	System Start-up Time (Wake-up from Condition Where f_{SYS} is Off)	—	$f_{SYS}=f_H\sim f_H/64$, $f_H=f_{HIRC}$	—	128	—	t_{SYS}
		—	$f_{SYS}=f_{SUB}=f_{LIRC}$	—	2	—	t_{SUB}
	System Start-up Time (Wake-up from Condition Where f_{SYS} is On)	—	$f_{SYS}=f_H\sim f_H/64$, $f_H=f_{HIRC}$	—	2	—	t_{SYS}
		—	$f_{SYS}=f_{SUB}=f_{LIRC}$	—	2	—	t_{SUB}
	System Speed Switch Time (FAST to SLOW Mode or SLOW to FAST Mode)	—	f_{HIRC} switches from off to on	—	128	—	t_{HIRC}
t_{RSTD}	System Reset Delay Time (Reset Source from Power-on Reset or LVR Hardware Reset)	—	RR_{POR}=5V/ms	5	16	80	ms
	System Reset Delay Time (Reset Source from WDT Overflow or $\overline{RES}$ Pin Reset)	—	—				

Note: 1. For the System Start-up time values, whether f_{SYS} is on or off depends upon the mode type and the chosen f_{SYS} system oscillator. Details are provided in the System Operating Modes section.

2. The time units, shown by the symbols, t_{HIRC}, etc., are the inverse of the corresponding frequency values as provided in the frequency tables. For example $t_{HIRC}=1/f_{HIRC}$, $t_{LIRC}=1/f_{LIRC}$, etc.

3. If the LIRC is used as the system clock and if it is off when in the SLEEP Mode, then an additional LIRC start up time, t_{START}, as provided in the LIRC frequency table, must be added to the t_{SST} time in the table above.

4. The System Speed Switch Time is effectively the time taken for the newly activated oscillator to start up.

Input/Output Characteristics

Ta=-40°C~85°C, unless otherwise specified

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
V_{IL}	Input Low Voltage for I/O Ports	5V	—	0	—	1.5	V
		—	—	0	—	$0.2V_{DD}$	
	Input Low Voltage for $\overline{RES}$ Pin	—	$V_{DD}\geq 2.7$	0	—	$0.4V_{DD}$	V
		—	$2.0\leq V_{DD}<2.7$	0	—	$0.3V_{DD}$	V
V_{IH}	Input High Voltage for I/O Ports	5V	—	3.5	—	5	V
		—	—	$0.8V_{DD}$	—	V_{DD}	
	Input High Voltage for $\overline{RES}$ Pin	—	—	$0.9V_{DD}$	—	V_{DD}	V
I_{OL}	Sink Current for I/O Ports	3V	$V_{OL}=0.1V_{DD}$	5	10	—	mA
		5V		10	20	—	
I_{OH}	Source Current for I/O Ports	3V	$V_{OH}=0.9V_{DD}$	-2.5	-5.0	—	mA
		5V		-5.0	-10	—	
R_{PH}	Pull-high Resistance for I/O Ports (Note)	3V	—	20	60	100	kΩ
		5V	—	10	30	50	
I_{LEAK}	Input Leakage Current for I/O Ports	5V	$V_{IN}=V_{DD}$ or $V_{IN}=V_{SS}$	—	—	±1	μA
t_{TC}	TCn Clock Input Minimum Pulse Width	—	—	20	—	—	ns
t_{INT}	Interrupt Input Pin Minimum Pulse Width	—	—	0.3	—	—	μs
t_{RES}	External Reset Minimum Low Pulse Width	—	—	0.3	—	—	μs

Note: The R_{PH} internal pull high resistance value is calculated by connecting to ground and enabling the input pin with a pull-high resistor and then measuring the pin current at the specified supply voltage level. Dividing the voltage by this measured current provides the R_{PH} value.

Memory Characteristics

Ta=-40°C~85°C, unless otherwise specified

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
V_{DD}	V_{DD} for Read – ORPP Memory	—	—	2.0	—	5.5	V
	V_{DD} for Write – ORPP Memory	—	—	4.5	—	5.5	V
OTP Program Memory							
V_{PP}	V_{PP} for Write – ORPP Memory	—	—	8.25	8.50	8.75	V
t_{WR}	Write Cycle Time – ORPP Memory	—	—	—	300	450	μs
E_P	Cell Endurance – ORPP Memory	—	—	1	—	—	W
t_{RETD}	ROM Data Retention Time	—	Ta=25°C	—	40	—	Year
Flash Program Memory – for BX66V004 only							
t_{ACTV}	ROM Activation Time – Wake-up from Power Down Mode	—	—	32	—	64	μs
RAM Data Memory							
V_{DR}	RAM Data Retention Voltage	—	—	1	—	—	V

Note: 1. "W" means Write times.

2. The ROM activation time t_{ACTV} should be added when calculating the total system start-up time of a wake-up from the IDLE/SLEEP mode.

A/D Converter Electrical Characteristics

Ta=-40°C~85°C

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
V_{ADI}	Input Voltage	—	—	0	—	V_{REF}	V
V_{REF}	Reference Voltage	—	—	2	—	V_{DD}	V
N_R	Resolution	—	—	–	—	8	Bit
DNL	Differential Nonlinearity	—	V_{REF}=V_{DD}, t_{ADCK}=0.5μs	-1	—	1	LSB
INL	Integral Nonlinearity	—	V_{REF}=V_{DD}, t_{ADCK}=0.5μs	-1	—	1	LSB
I_{ADC}	Additional Current for A/D Converter Enable	2.2V	No load (t_{ADCK}=0.5μs)	—	150	225	μA
		3V		—	220	330	
		5V		—	400	600	
t_{ADCK}	Clock Period	—	—	0.5	—	10.0	μs
t_{ON2ST}	A/D Converter On-to-Start Time	—	—	4	—	—	μs
t_{ADC}	Conversion Time (Includes A/D Sample and Hold Time)	—	—	–	12	—	t_{ADCK}

LVR Electrical Characteristics

Ta=-40°C~85°C

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
V_{LVR}	Low Voltage Reset Voltage	—	LVR enable, voltage select 1.9V	-5%	1.9	+5%	V
			LVR enable, voltage select 2.1V	-5%	2.1	+5%	
			LVR enable, voltage select 3.15V	-5%	3.15	+5%	
			LVR enable, voltage select 4.2V	-5%	4.2	+5%	
I_{LVR}	Operating Current	3V	LVR enable, V_{LVR}=1.9V	—	—	15	μA
		5V	LVR enable, V_{LVR}=1.9V	—	15	30	
t_{LVR}	Minimum Low Voltage Width to Reset	—	—	120	240	480	μs

LCD Electrical Characteristics

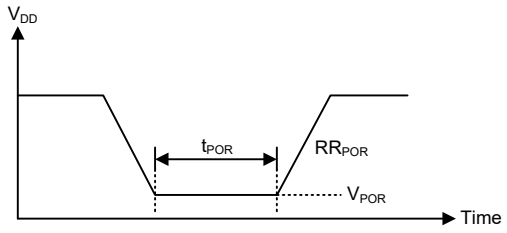
Ta=-40°C~85°C

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
I_{BIAS}	V_{DD}/2 Bias Current for LCD	3V	ISEL[2:0]=000B	10.5	15.0	22.5	μA
		5V		17.5	25.0	34.5	μA
		3V	ISEL[2:0]=001B	21	30	39	μA
		5V		35	50	65	μA
		3V	ISEL[2:0]=010B	42	60	78	μA
		5V		70	100	130	μA
		3V	ISEL[2:0]=011B	82.6	118.0	153.4	μA
		5V		140	200	260	μA
		3V	ISEL[2:0]=100B	1.5	3.0	4.5	μA
		5V		2.5	5.0	7.5	μA
		3V	ISEL[2:0]=101B~111B	5.2	7.5	9.8	μA
		5V		8.7	12.5	16.3	μA
V_{SCOM}	V_{DD}/2 Voltage for LCD COM Port	2.2V ~ 5.5V	No load	0.475V_{DD}	0.500V_{DD}	0.525V_{DD}	V

Power-on Reset Characteristics

Ta=-40°C~85°C

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
V_{POR}	V_{DD} Start Voltage to Ensure Power-on Reset	—	—	—	—	100	mV
RR_{POR}	V_{DD} Rising Rate to Ensure Power-on Reset	—	—	0.035	—	—	V/ms
t_{POR}	Minimum Time for V_{DD} Stays at V_{POR} to Ensure Power-on Reset	—	—	1	—	—	ms



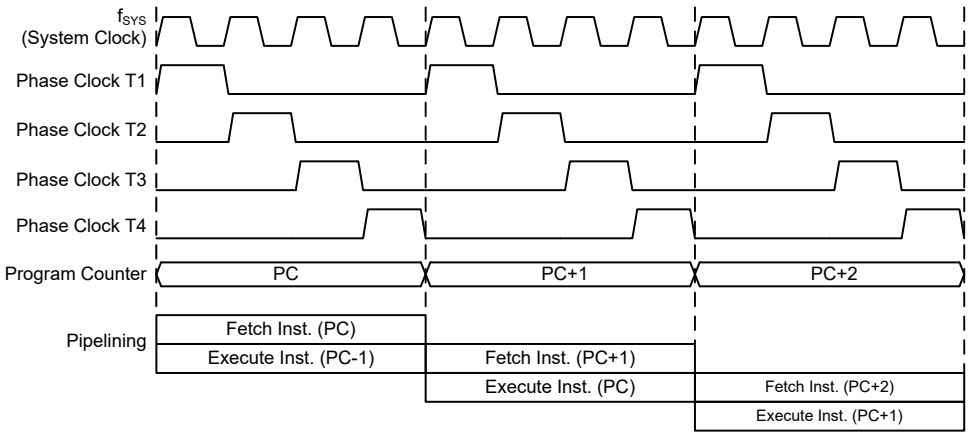
System Architecture

A key factor in the high-performance features of the microcontrollers is attributed to their internal system architecture. The device takes advantage of the usual features found within RISC microcontrollers providing increased speed of operation and enhanced performance. The pipelining scheme is implemented in such a way that instruction fetching and instruction execution are overlapped, hence instructions are effectively executed in one cycle, with the exception of branch or call instructions which need one more cycle. An 8-bit wide ALU is used in practically all instruction set operations, which carries out arithmetic operations, logic operations, rotation, increment, decrement, branch decisions, etc. The internal data path is simplified by moving data through the Accumulator and the ALU. Certain internal registers are implemented in the Data Memory and can be directly or indirectly addressed. The simple addressing methods of these registers along with additional architectural features ensure that a minimum of external components is required to provide a functional I/O control system with maximum reliability and flexibility. This makes the device suitable for affordable, high-volume production for controller applications.

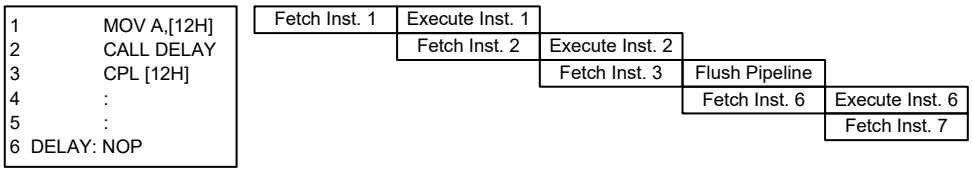
Clocking and Pipelining

The main system clock, derived from either a HIRC or LIRC oscillator is subdivided into four internally generated non-overlapping clocks, T1~T4. The Program Counter is incremented at the beginning of the T1 clock during which time a new instruction is fetched. The remaining T2~T4 clocks carry out the decoding and execution functions. In this way, one T1~T4 clock cycle forms one instruction cycle. Although the fetching and execution of instructions takes place in consecutive instruction cycles, the pipelining structure of the microcontroller ensures that instructions are effectively executed in one instruction cycle. The exception to this are instructions where the contents of the Program Counter are changed, such as subroutine calls or jumps, in which case the instruction will take one more instruction cycle to execute.

For instructions involving branches, such as jump or call instructions, two machine cycles are required to complete instruction execution. An extra cycle is required as the program takes one cycle to first obtain the actual jump or call address and then another cycle to actually execute the branch. The requirement for this extra cycle should be taken into account by programmers in timing sensitive applications.



System Clocking and Pipelining



Instruction Fetching

Program Counter

During program execution, the Program Counter is used to keep track of the address of the next instruction to be executed. It is automatically incremented by one each time an instruction is executed except for instructions, such as "JMP" or "CALL" that demand a jump to a non-consecutive Program Memory address. Only the lower 8 bits, known as the Program Counter Low Register, are directly addressable by the application program.

When executing instructions requiring jumps to non-consecutive addresses such as a jump instruction, a subroutine call, interrupt or reset, etc., the microcontroller manages program control by loading the required address into the Program Counter. For conditional skip instructions, once the condition has been met, the next instruction, which has already been fetched during the present instruction execution, is discarded and a dummy cycle takes its place while the correct instruction is obtained.

Program Counter	
Program Counter High Byte	**PCL Register**
PC10~PC8	PCL7~PCL0

Program Counter

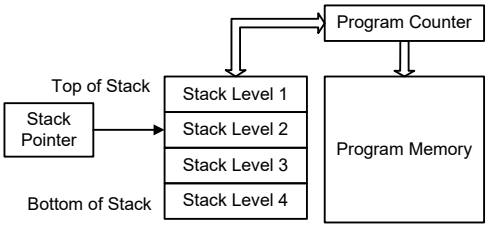
The lower byte of the Program Counter, known as the Program Counter Low register or PCL, is available for program control and is a readable and writeable register. By transferring data directly into this register, a short program jump can be executed directly; however, as only this low byte is available for manipulation, the jumps are limited to the present page of memory that is 256 locations. When such program jumps are executed it should also be noted that a dummy cycle will be inserted. Manipulating the PCL register may cause program branching, so an extra cycle is needed to pre-fetch.

Stack

This is a special part of the memory which is used to save the contents of the Program Counter only. The stack, organized into four levels, is neither part of the data nor part of the program space, and is neither readable nor writeable. The activated level is indexed by the Stack Pointer, and is neither readable nor writeable. At a subroutine call or interrupt acknowledge signal, the contents of the Program Counter are pushed onto the stack. At the end of a subroutine or an interrupt routine, signaled by a return instruction, RET or RETI, the Program Counter is restored to its previous value from the stack. After a device reset, the Stack Pointer will point to the top of the stack.

If the stack is full and an enabled interrupt takes place, the interrupt request flag will be recorded but the acknowledge signal will be inhibited. When the Stack Pointer is decremented, by RET or RETI, the interrupt will be serviced. This feature prevents stack overflow allowing the programmer to use the structure more easily. However, when the stack is full, a CALL subroutine instruction can still be executed which will result in a stack overflow. Precautions should be taken to avoid such cases which might cause unpredictable program branching.

If the stack is overflow, the first Program Counter save in the stack will be lost.



Arithmetic and Logic Unit – ALU

The arithmetic-logic unit or ALU is a critical area of the microcontroller that carries out arithmetic and logic operations of the instruction set. Connected to the main microcontroller data bus, the ALU receives related instruction codes and performs the required arithmetic or logical operations after which the result will be placed in the specified register. As these ALU calculation or operations may result in carry, borrow or other status changes, the status register will be correspondingly updated to reflect these changes. The ALU supports the following functions:

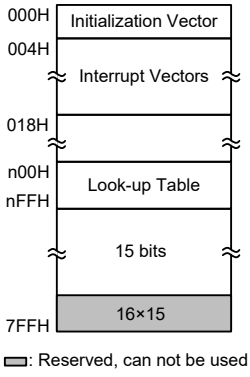
- Arithmetic operations: ADD, ADDM, ADC, ADCM, SUB, SUBM, SBC, SBCM, DAA
- Logic operations: AND, OR, XOR, ANDM, ORM, XORM, CPL, CPLA
- Rotation: RRA, RR, RRCA, RRC, RLA, RL, RLCA, RLC
- Increment and Decrement: INCA, INC, DECA, DEC
- Branch decision: JMP, SZ, SZA, SNZ, SIZ, SDZ, SIZA, SDZA, CALL, RET, RETI

OTP Program Memory

The Program Memory is the location where the user code or program is stored. The device is supplied with One-Time Programmable, OTP memory where users can program their application code into the device.

Structure

The Program Memory has a capacity of (2K-16)×15 bits. Note that the subtractive 16×15 bits space is reserved and cannot be used. The Program Memory is addressed by the Program Counter and also contains data, table information and interrupt entries. Table data, which can be set in any location within the Program Memory, is addressed by a separate table pointer register.



Program Memory Structure

Special Vectors

Within the Program Memory, certain locations are reserved for the reset and interrupts. The location 000H is reserved for use by the device reset for program initialisation. After a device reset is initiated, the program will jump to this location and begin execution.

Look-up Table

Any location within the Program Memory can be defined as a look-up table where programmers can store fixed data. To use the look-up table, the table pointer must first be configured by placing the address of the look up data to be retrieved in the table pointer register, TBLP. This register defines the total address of the look-up table.

After setting up the table pointer, the table data can be retrieved from the Program Memory using the "TABRD [m]" or "TABRDL[m]" instructions respectively. When the instruction is executed, the lower order table byte from the Program Memory will be transferred to the user defined Data Memory register [m] as specified in the instruction. The higher order table data byte from the Program Memory will be transferred to the TBLH special register. Any unused bits in this transferred higher order byte will be read as "0".

The accompanying diagram illustrates the addressing data flow of the look-up table.

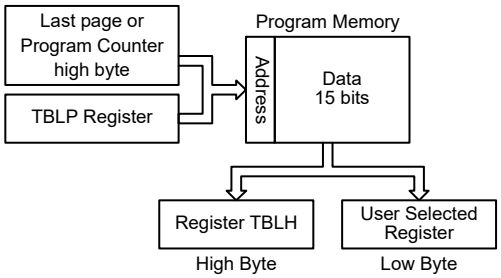


Table Program Example

The following example shows how the table pointer and table data is defined and retrieved from the microcontrollers. This example uses raw table data located in the Program Memory which is stored there using the ORG statement. The value at this ORG statement is "0700H" which refers to the start address of the last page within the 2K words Program Memory of the device. The table pointer low byte register is set here to have an initial value of "06H". This will ensure that the first data read from the data table will be at the Program Memory address "0706H" or 6 locations after the start of the last page. Note that the value for the table pointer is referenced to the specific address of the TBLP register if the "TABRD [m]" instruction is being used. The high byte of the table data which in this case is equal to zero will be transferred to the TBLH register automatically when the "TABRD [m] instruction is executed.

Because the TBLH register is a read-only register and cannot be restored, care should be taken to ensure its protection if both the main routine and Interrupt Service Routine use table read instructions. If using the table read instructions, the Interrupt Service Routines may change the value of the TBLH and subsequently cause errors if used again by the main routine. As a rule it is recommended that simultaneous use of the table read instructions should be avoided. However, in situations where simultaneous use cannot be avoided, the interrupts should be disabled prior to the execution of any main routine table-read instructions. Note that all table related instructions require two instruction cycles to complete their operation.

Table Read Program Example

```
tempreg1 db ?      ; temporary register #1
tempreg2 db ?      ; temporary register #2
:
:
mov a,06h          ; initialise low table pointer - note that this address is referenced
mov tblp,a         ; to the last page or present page
:
:
tabrd tempreg1     ; transfers value in table referenced by table pointer,
                   ; data at program memory address "0706H" transferred to tempreg1 and TBLH
dec tblp           ; reduce value of table pointer by one
tabrd tempreg2     ; transfers value in table referenced by table pointer,
                   ; data at program memory address "0705H" transferred to tempreg2 and TBLH
                   ; in this example the data "1AH" is transferred to tempreg1 and data "0FH"
                   ; to register tempreg2
                   ; the value "00H" will be transferred to the high byte register TBLH
:
:
org 0700h          ; sets initial address of program memory
dc 00Ah, 00Bh, 00Ch, 00Dh, 00Eh, 00Fh, 01Ah, 01Bh
```

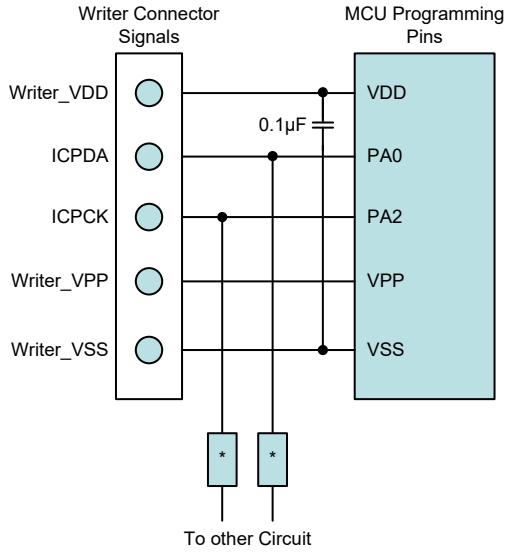
In Circuit Programming – ICP

The provision of OTP type Program Memory, users can program their application One-Time into the device. As an additional convenience, The device has provided a means of programming in-circuit using a 5-pin interface. This provides manufacturers with the possibility of manufacturing their circuit boards complete with an un-programmed microcontroller, and then programming the program at a later stage.

Writer Pins	MCU Programming Pins	Pin Description
ICPDA	PA0	Programming Serial Data/Address
ICPCK	PA2	Programming Clock
VPP	VPP	Programming OTP ROM power supply (8.5V)
VDD	VDD	Power Supply. A 0.1μF capacitor is required to be connected between VDD and VSS for programming.
VSS	VSS	Ground

The Program Memory can be programmed serially in-circuit using this 5-wire interface. Data is downloaded and uploaded serially on a single pin with an additional line for the clock. Three additional lines are required for the power supply. The technical details regarding the in-circuit programming of the device is beyond the scope of this document and will be supplied in supplementary literature.

During the programming process, the user must take care of the ICPDA and ICPCK pins for data and clock programming purposes to ensure that no other outputs are connected to these two pins.



Note: 1. A 0.1μF capacitor is required to be connected between VDD and VSS for ICP programming and located as close to these pins as possible.

2. * may be resistor or capacitor. The resistance of * must be greater than 1kΩ or the capacitance of * must be less than 1nF.

On-Chip Debug Support – OCDS

There is an EV chip named BX66V004 which is used to emulate the real MCU device named BX66R004. The EV chip device also provides an "On-Chip Debug" function to debug the real MCU device during the development process. The EV chip and the real MCU device are almost functionally compatible except for "On-Chip Debug" function. Users can use the EV chip device to emulate the real chip device behavior by connecting the OCDSDA and OCDSCK pins to the ICE development tools. The OCDSDA pin is the OCDS Data/Address input/output pin while the OCDSCK pin is the OCDS clock input pin. When users use the EV chip for debugging, other functions which are shared with the OCDSDA and OCDSCK pins in the device will have no effect in the EV chip.

ICE Pins	EV Chip Pins	Pin Description
OCDSDA	OCDSDA	On-Chip Debug Support Data/Address input/output
OCDSCK	OCDSCK	On-Chip Debug Support Clock input
VDD	VDD	Power Supply
VSS	VSS	Ground

OTP ROM Parameter Program – ORPP

This device contains an ORPP function. The provision of the ORPP function offers users the convenience of OTP Memory programming features. Note that the Write operation only writes data to the last page of ROM, and the data can only be written once and cannot be erased.

Before the write operation is implemented, the VPP pin must be connected to an 8.5V power and after the write operation is completed, the high voltage power should be removed from the VPP pin. If the VPP function is pin-shared with an I/O port, the corresponding I/O port cannot be set as an output when it is used as the VPP function.

ORPP Registers

Three registers control the overall operation of the internal ORPP function. These are data registers ODL and ODH, and a control register OCR.

Register Name	Bit							
	7	6	5	4	3	2	1	0
OCR	—	—	—	—	WREN	WR	—	—
ODL	D7	D6	D5	D4	D3	D2	D1	D0
ODH	—	D14	D13	D12	D11	D10	D9	D8

ORPP Register List

• ODL Register

Bit	7	6	5	4	3	2	1	0
Name	D7	D6	D5	D4	D3	D2	D1	D0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	0	0	0	0	0

Bit 7~0 **D7~D0**: ORPP program memory data bit 7 ~ bit 0

• ODH Register

Bit	7	6	5	4	3	2	1	0
Name	—	D14	D13	D12	D11	D10	D9	D8
R/W	—	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	—	0	0	0	0	0	0	0

Bit 7 Unimplemented, read as "0"

Bit 6~0 **D14~D8**: ORPP program memory data bit 14 ~ bit 8

• OCR Register

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	WREN	WR	—	—
R/W	—	—	—	—	R/W	R/W	—	—
POR	—	—	—	—	0	0	—	—

Bit 7~4 Unimplemented, read as "0"

Bit 3 **WREN**: ORPP Write Enable
0: Disable
1: Enable
This is the ORPP Write Enable Bit which must be set high before write operations are carried out. This bit will be automatically reset to zero by the hardware after the write cycle has finished. Clearing this bit to zero will inhibit ORPP write operations.

Bit 2 **WR**: ORPP Write Control
0: Write cycle has finished
1: Activate a write cycle
This is the ORPP Write Control Bit and when set high by the application program will activate a write cycle. This bit will be automatically reset to zero by the hardware after the write cycle has finished. Setting this bit high will have no effect if the WREN has not first been set high.

Bit 1~0 Unimplemented, read as "0"

Note: 1. The WREN and WR cannot be set high at the same time in one instruction.
2. Note that the CPU will be stopped when a write operation is successfully activated.
3. Ensure that the f_{SUB} clock is stable before executing the write operation.
4. Ensure that the write operation is totally complete before executing other operations.

ORPP Writing Data to the OTP Program Memory

For ORPP write operation the data to be written should be placed in the ODH and ODL registers and the desired write address should first be placed in the TBLP register. To write data to ROM, the write enable bit, WREN, in the OCR register must first be set high to enable the write function. After this, the WR bit in the OCR register must be immediately set high to initiate a write cycle. These two instructions must be executed in two consecutive instruction cycles to activate a write operation successfully. The global interrupt bit EMI should also first be cleared before implementing any write operations, and then set again after a valid write activation procedure has completed. Note that the CPU will be stopped when a write operation is successfully activated. When the write cycle terminates, the CPU will resume executing the application program. And the WR bit will be automatically cleared to zero by the microcontroller, informing the user that the data has been written to the OTP Program Memory.

ORPP Reading Data from the OTP Program Memory

For ORPP read operation the desired address should first be placed in the TBLP register. Then the data can be retrieved from the program memory using the "TABRDL [m]" instruction. When the instruction is executed, the lower order table byte from the Program Memory will be transferred to the user defined Data Memory register [m] as specified in the instruction. The higher order table data byte from the Program Memory will be transferred to the TBLH special register.

Programming Considerations

Care must be taken that data is not inadvertently written to ROM. Protection can be enhanced by ensuring that the Write Enable bit is normally cleared to zero when not writing. Although certainly not necessary, consideration might be given in the application program to the checking of the validity of new write data by a simple read back process.

When writing data the WR bit must be set high immediately after the WREN bit has been set high, to ensure the write cycle executes correctly. The global interrupt bit EMI should also be cleared before a write cycle is executed and then set high again after a write activation procedure has completed. Note that the device should not enter the IDLE or SLEEP mode until ORPP write operation is totally complete. Otherwise, ORPP write operation will fail.

Programming Examples

ORPP Reading data from the OTP Program Memory

```
Tempreg1  db ?          ; temporary register
MOV  A, 03H
MOV  TBLP, A            ; set read address 03H
TABRDL Tempreg1         ; transfers value in table (last page) referenced by table pointer,
                        ; data at program memory address "0703H" transferred to tempreg1 and TBLH
```

ORPP Writing Data to the OTP Program Memory

```
MOV  A, ORPP_ADRES      ; user defined address
MOV  TBLP, A
MOV  A, ORPP _DATA_L    ; user defined data
MOV  ODL, A
MOV  A, ORPP _DATA_H
MOV  ODH, A
MOV  A, 00H
MOV  OCR, A
CLR  EMI
SET  WREN               ; set WREN bit, enable write operation
SET  WR                 ; start Write Cycle - set WR bit - executed immediately
                        ; after setting WREN bit
SET  EMI
BACK:
SZ   WR                 ; check for write cycle end
JMP  BACK
NOP
```

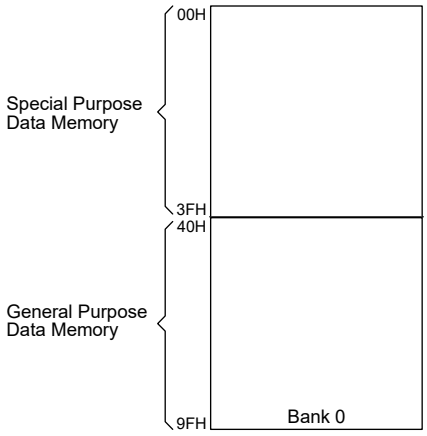
Data Memory

The Data Memory is a volatile area of 8-bit wide RAM internal memory and is the location where temporary information is stored.

Structure

Categorised into two types, the first of these is an area of RAM, known as the Special Function Data Memory. These registers have fixed locations and are necessary for correct operation of the device. Many of these registers can be read from and written to directly under program control, however, some remain protected from user manipulation. The second area of Data Memory is known as the General Purpose Data Memory, which is reserved for general purpose use. All locations within this area are read and write accessible under program control.

The start address of the Data Memory for the device is 00H. The address range of the Special Purpose Data Memory for the device is from 00H to 3FH while the address range of the General Purpose Data Memory is from 40H to 9FH.



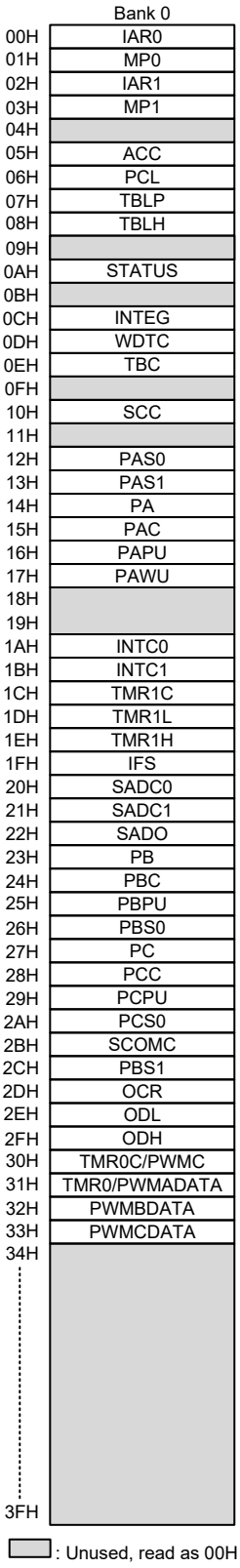
Data Memory Structure

General Purpose Data Memory

All microcontroller programs require an area of read/write memory where temporary data can be stored and retrieved for use later. It is this area of RAM memory that is known as General Purpose Data Memory. This area of Data Memory is fully accessible by the user programming for both reading and writing operations. By using the bit operation instructions individual bits can be set or reset under program control giving the user a large range of flexibility for bit manipulation in the Data Memory.

Special Purpose Data Memory

This area of Data Memory is where registers, necessary for the correct operation of the microcontroller, are stored. Most of the registers are both readable and writeable but some are protected and are readable only, the details of which are located under the relevant Special Function Register section. Note that for locations that are unused, any read instruction to these addresses will return the value “00H”.



Special Purpose Data Memory

Special Function Register Description

Most of the Special Function Register details will be described in the relevant functional section; however several registers require a separate description in this section.

Indirect Addressing Registers – IAR0, IAR1

The Indirect Addressing Registers, IAR0 and IAR1, although having its location in normal RAM register space, do not actually physically exist as normal registers. The method of indirect addressing for RAM data manipulation uses this Indirect Addressing Registers and Memory Pointers, in contrast to direct memory addressing, where the actual memory address is specified. Actions on the IAR0 and IAR1 registers will result in no actual read or write operation to these registers but rather to the memory location specified by the corresponding Memory Pointers, MP0 or MP1. Acting as a pair, IAR0 and MP0 or IAR1 and MP1 can together access data from Bank 0. As the Indirect Addressing Register is not physically implemented, reading the Indirect Addressing Register will return a result of "00H" and writing to the register will result in no operation.

Memory Pointers – MP0, MP1

Two Memory Pointers, known as MP0 and MP1 are provided. These Memory Pointers are physically implemented in the Data Memory and can be manipulated in the same way as normal registers providing a convenient way with which to address and track data. When any operation to the Indirect Addressing Register is carried out, the actual address that the microcontroller is directed to is the address specified by the Memory Pointer. MP0 and MP1, together with the Addressing Registers, IAR0 and IAR1, all can be used to access data from Bank 0.

The following example shows how to clear a section of four Data Memory locations already defined as locations adres1 to adres4.

Indirect Addressing Program Example

```
data .section 'data'
adres1   db ?
adres2   db ?
adres3   db ?
adres4   db ?
block    db ?
code .section at 0 'code'
org 00h
start:
     mov a, 04h           ; set size of block
     mov block, a
     mov a, offset adres1  ; Accumulator loaded with first RAM address
     mov MP0, a           ; set memory pointer with first RAM address
loop:
     clr IAR0             ; clear the data at address defined by MP0
     inc MP0              ; increase memory pointer
     sdz block            ; check if last memory location has been cleared
     jmp loop
continue:
```

The important point to note here is that in the examples shown above, no reference is made to specific Data Memory addresses.

Accumulator – ACC

The Accumulator is central to the operation of any microcontroller and is closely related with operations carried out by the ALU. The Accumulator is the place where all intermediate results from the ALU are stored. Without the Accumulator it would be necessary to write the result of each calculation or logical operation such as addition, subtraction, shift, etc., to the Data Memory resulting in higher programming and timing overheads. Data transfer operations usually involve the temporary storage function of the Accumulator; for example, when transferring data between one user-defined register and another, it is necessary to do this by passing the data through the Accumulator as no direct transfer between two registers is permitted.

Program Counter Low Byte Register – PCL

To provide additional program control functions, the low byte of the Program Counter is made accessible to programmers by locating it within the Special Purpose area of the Data Memory. By manipulating this register, direct jumps to other program locations are easily implemented. Loading a value directly into this PCL register will cause a jump to the specified Program Memory location, however, as the register is only 8-bit wide, only jumps within the current Program Memory page are permitted. When such operations are used, note that a dummy cycle will be inserted.

Look-up Table Registers – TBLP, TBLH

The special function register TBLP is used to control operation of the look-up table which is stored in the Program Memory. TBLP is the table pointer and indicates the location where the table data is located. Its value must be set before any table read commands are executed. Its value can be changed, for example using the “INC” or “DEC” instructions, allowing for easy table data pointing and reading. TBLH is the location where the high order byte of the table data is stored after a table read data instruction has been executed. Note that the lower order table data byte is transferred to a user defined location.

Status Register – STATUS

This 8-bit register contains the zero flag (Z), carry flag (C), auxiliary carry flag (AC), overflow flag (OV), power down flag (PDF), and watchdog time-out flag (TO). These arithmetic/logical operation and system management flags are used to record the status and operation of the microcontroller.

With the exception of the TO and PDF flags, bits in the status register can be altered by instructions like most other registers. Any data written into the status register will not change the TO or PDF flag. In addition, operations related to the status register may give different results due to the different instruction operations. The TO flag can be affected only by a system power-up, a WDT time-out or by executing the “CLR WDT” or “HALT” instruction. The PDF flag is affected only by executing the “HALT” or “CLR WDT” instruction or during a system power-up.

The Z, OV, AC and C flags generally reflect the status of the latest operations.

- C is set if an operation results in a carry during an addition operation or if a borrow does not take place during a subtraction operation; otherwise C is cleared. C is also affected by a rotate through carry instruction.
- AC is set if an operation results in a carry out of the low nibbles in addition, or no borrow from the high nibble into the low nibble in subtraction; otherwise AC is cleared.
- Z is set if the result of an arithmetic or logical operation is zero; otherwise Z is cleared.
- OV is set if an operation results in a carry into the highest-order bit but not a carry out of the highest-order bit, or vice versa; otherwise OV is cleared.

- PDF is cleared by a system power-up or executing the “CLR WDT” instruction. PDF is set by executing the “HALT” instruction.
- TO is cleared by a system power-up or executing the “CLR WDT” or “HALT” instruction. TO is set by a WDT time-out.

In addition, on entering an interrupt sequence or executing a subroutine call, the status register will not be pushed onto the stack automatically. If the contents of the status registers are important and if the subroutine can corrupt the status register, precautions must be taken to correctly save it.

• STATUS Register

Bit	7	6	5	4	3	2	1	0
Name	—	—	TO	PDF	OV	Z	AC	C
R/W	—	—	R	R	R/W	R/W	R/W	R/W
POR	—	—	0	0	x	x	x	x

“x”: unknown

Bit 7~6 Unimplemented, read as “0”

Bit 5 **TO**: Watchdog Time-out flag
0: After power up or executing the “CLR WDT” or “HALT” instruction
1: A watchdog time-out occurred

Bit 4 **PDF**: Power down flag
0: After power up or executing the “CLR WDT” instruction
1: By executing the “HALT” instruction

Bit 3 **OV**: Overflow flag
0: No overflow
1: An operation results in a carry into the highest-order bit but not a carry out of the highest-order bit or vice versa

Bit 2 **Z**: Zero flag
0: The result of an arithmetic or logical operation is not zero
1: The result of an arithmetic or logical operation is zero

Bit 1 **AC**: Auxiliary flag
0: No auxiliary carry
1: An operation results in a carry out of the low nibbles in addition, or no borrow from the high nibble into the low nibble in subtraction

Bit 0 **C**: Carry flag
0: No carry-out
1: An operation results in a carry during an addition operation or if a borrow does not take place during a subtraction operation

The “C” flag is also affected by a rotate through carry instruction.

Oscillators

Various oscillator options offer the user a wide range of functions according to their various application requirements. The flexible features of the oscillator functions ensure that the best optimisation can be achieved in terms of speed and power saving. Oscillator operations are selected through the relevant control registers.

Oscillator Overview

In addition to being the source of the main system clock the oscillators also provide clock sources for the Watchdog Timer and Time Base Interrupt. The fully integrated internal oscillators, requiring no external components, are provided to form a wide range of both fast and slow system oscillators. The higher frequency oscillator provides higher performance but carry with it the disadvantage of

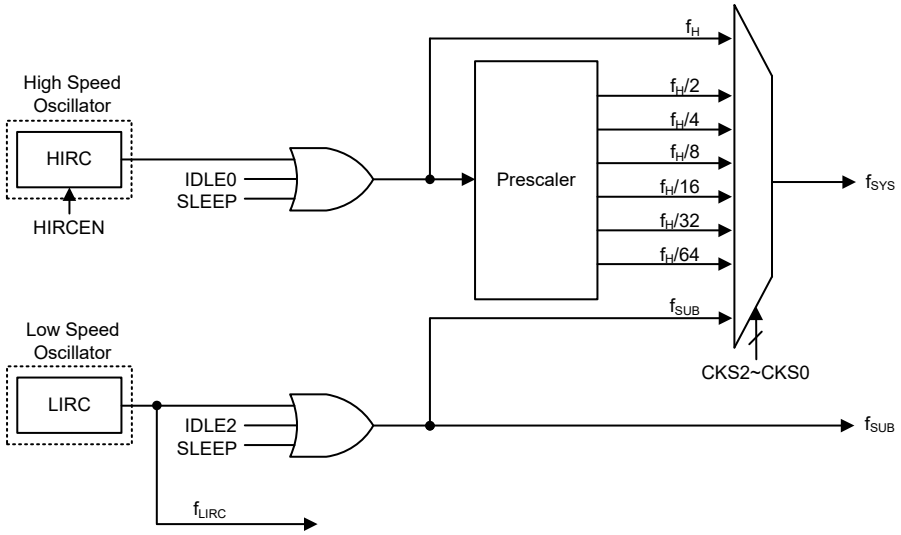
higher power requirements, while the opposite is of course true for the lower frequency oscillator. With the capability of dynamically switching between fast and slow system clock, the device has the flexibility to optimize the performance/power ratio, a feature especially important in power sensitive portable applications.

Type	Name	Frequency
Internal High Speed RC	HIRC	8MHz
Internal Low Speed RC	LIRC	32kHz

Oscillator Types

System Clock Configurations

There are two oscillator sources, one high speed oscillator and one low speed oscillator. The high speed system clock is sourced from the internal 8MHz RC oscillator, HIRC. The low speed oscillator is the internal 32kHz RC oscillator, LIRC. Selecting whether the low or high speed oscillator is used as the system oscillator is implemented using the CKS2~CKS0 bits in the SCC register and the system clock can be dynamically selected.



System Clock Configuration

Internal High Speed RC Oscillator – HIRC

The internal RC oscillator is a fully integrated system oscillator requiring no external components. The internal RC oscillator has one fixed frequency of 8MHz. Device trimming during the manufacturing process and the inclusion of internal frequency compensation circuits are used to ensure that the influence of the power supply voltage, temperature and process variations on the oscillation frequency are minimised. Note that this internal system clock option requires no external pins for its operation.

Internal 32kHz Oscillator – LIRC

The Internal 32kHz System Oscillator is a fully integrated low frequency RC oscillator with a typical frequency of 32kHz, requiring no external components for its implementation. Device trimming during the manufacturing process and the inclusion of internal frequency compensation circuits are used to ensure that the influence of the power supply voltage, temperature and process variations on the oscillation frequency are minimised.

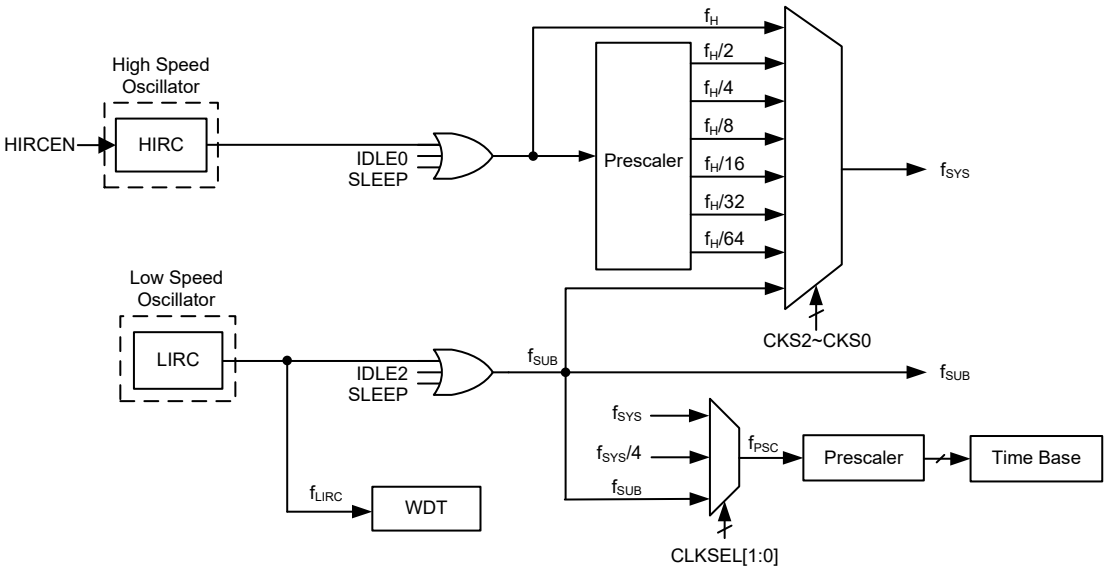
Operating Modes and System Clocks

Present day applications require that their microcontrollers have high performance but often still demand that they consume as little power as possible, conflicting requirements that are especially true in battery powered portable applications. The fast clocks required for high performance will by their nature increase current consumption and of course vice versa, lower speed clocks reduce current consumption. As the device has provided both high and low speed clock sources and the means to switch between them dynamically, the user can optimise the operation of their microcontroller to achieve the best performance/power ratio.

System Clocks

The device has many different clock sources for both the CPU and peripheral function operation. By providing the user with a wide range of clock options using register programming, a clock system can be configured to obtain maximum application performance.

The main system clock, can come from either a high frequency, f_H, or low frequency, f_{SUB}, source, and is selected using the CKS2~CKS0 bits in the SCC register. The high speed system clock is sourced from the HIRC oscillator. The low speed system clock source is sourced from the LIRC oscillator. The other choice, which is a divided version of the high speed system oscillator has a range of $f_H/2$~$f_H/64$.



Device Clock Configurations

Note: When the system clock source f_{SYS} is switched to f_{SUB} from f_H, the high speed oscillator will stop to conserve the power or continue to oscillate to provide the clock source, f_H~$f_H/64$, for peripheral circuit to use, which is determined by configuring the corresponding high speed oscillator enable control bit.

System Operation Modes

There are six different modes of operation for the microcontrollers, each one with its own special characteristics and which can be chosen according to the specific performance and power requirements of the application. There are two modes allowing normal operation of the microcontroller, the FAST Mode and SLOW Mode. The remaining four modes, the SLEEP, IDLE0, IDLE1 and IDLE2 Mode are used when the microcontroller CPU is switched off to conserve power.

Operation Mode	CPU	Register Setting			f_{SYS}	f_H	f_{SUB}	f_{LIRC}
		FHIDEN	FSIDEN	CKS2~CKS0				
FAST	On	x	x	000~110	f_H~f_H/64	On	On	On
SLOW	On	x	x	111	f_{SUB}	On/Off (1)	On	On
IDLE0	Off	0	1	000~110	Off	Off	On	On
				111	On			
IDLE1	Off	1	1	xxx	On	On	On	On
IDLE2	Off	1	0	000~110	On	On	Off	On
				111	Off			
SLEEP	Off	0	0	xxx	Off	Off	Off	On/Off (2)

"x": Don't care

Note: 1. The f_H clock will be switched on or off by configuring the corresponding oscillator enable bit in the SLOW mode.

2. The f_{LIRC} clock can be switched on or off which is controlled by the WDT function being enabled or disabled in the SLEEP mode.

FAST Mode

This is one of the main operating modes where the microcontrollers have all of their functions operational and where the system clock is provided by the high speed oscillator. This mode operates allowing the microcontrollers to operate normally with a clock source which will come from the high speed oscillator, HIRC. The high speed oscillator will however first be divided by a ratio ranging from 1 to 64, the actual ratio being selected by the CKS2~CKS0 bits in the SCC register. Although a high speed oscillator is used, running the microcontrollers at a divided clock ratio reduces the operating current.

SLOW Mode

This is also a mode where the microcontroller operates normally although now with a slower speed clock source. The clock source used will be from f_{SUB}, which is derived from the LIRC oscillator.

SLEEP Mode

The SLEEP Mode is entered when an HALT instruction is executed and when the FHIDEN and FSIDEN bit are low. In the SLEEP mode the CPU will be stopped. The f_{SUB} clock provided to the peripheral function will also be stopped, too. However the f_{LIRC} clock can continues to operate if the WDT function is enabled.

IDLE0 Mode

The IDLE0 Mode is entered when an HALT instruction is executed and when the FHIDEN bit in the SCC register is low and the FSIDEN bit in the SCC register is high. In the IDLE0 Mode the CPU will be switched off but the low speed oscillator will be turned on to drive some peripheral functions.

IDLE1 Mode

The IDLE1 Mode is entered when an HALT instruction is executed and when the FHIDEN bit in the SCC register is high and the FSIDEN bit in the SCC register is high. In the IDLE1 Mode the CPU will be switched off but both the high and low speed oscillators will be turned on to provide a clock source to keep some peripheral functions operational.

IDLE2 Mode

The IDLE2 Mode is entered when an HALT instruction is executed and when the FHIDEN bit in the SCC register is high and the FSIDEN bit in the SCC register is low. In the IDLE2 Mode the CPU will be switched off but the high speed oscillator will be turned on to provide a clock source to keep some peripheral functions operational.

Control Register

The SCC register is used to control the system clock and the corresponding oscillator configurations.

• **SCC Register**

Bit	7	6	5	4	3	2	1	0
Name	CKS2	CKS1	CKS0	—	HIRCF	HIRCEN	FHIDEN	FSIDEN
R/W	R/W	R/W	R/W	—	R	R/W	R/W	R/W
POR	1	1	1	—	0	0	0	0

Bit 7~5 **CKS2~CKS0**: System clock selection
000: f_H
001: $f_H/2$
010: $f_H/4$
011: $f_H/8$
100: $f_H/16$
101: $f_H/32$
110: $f_H/64$
111: f_{SUB}
These three bits are used to select which clock is used as the system clock source. In addition to the system clock source directly derived from f_H or f_{SUB}, a divided version of the high speed system oscillator can also be chosen as the system clock source.

Bit 4 Unimplemented, read as "0"

Bit 3 **HIRCF**: HIRC oscillator stable flag
0: HIRC unstable
1: HIRC stable
This bit is used to indicate whether the HIRC oscillator is stable or not. When the HIRCEN bit is set to 1 to enable the HIRC oscillator or the HIRC frequency selection is changed by the application program, the HIRCF bit will first be cleared to 0 and then set to 1 after the HIRC oscillator is stable.

Bit 2 **HIRCEN**: HIRC oscillator enable control
0: Disable
1: Enable

Bit 1 **FHIDEN**: High Frequency oscillator control when CPU is switched off
0: Disable
1: Enable
This bit is used to control whether the high speed oscillator is activated or stopped when the CPU is switched off by executing an "HALT" instruction.

Bit 0 **FSIDEN**: Low Frequency oscillator control when CPU is switched off
0: Disable
1: Enable
This bit is used to control whether the low speed oscillator is activated or stopped when the CPU is switched off by executing an "HALT" instruction.

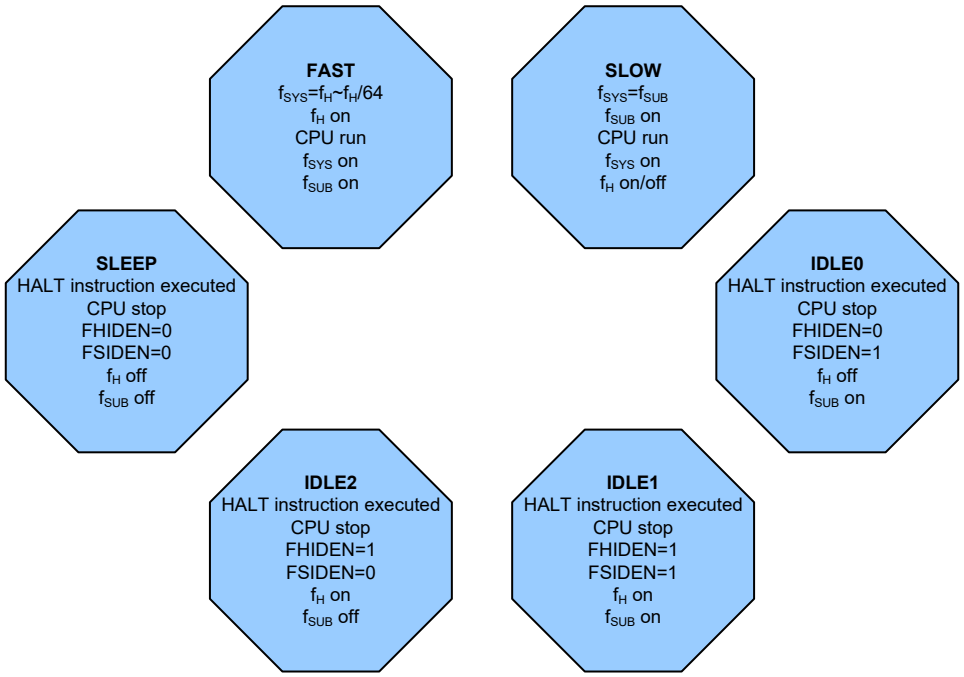
Note: A certain delay is required before the relevant clock is successfully switched to the target clock source after any clock switching setup using the CKS2~CKS0 bits. A proper delay time must be arranged before executing the following operations which require immediate reaction with the target clock source.

Clock switching delay time = $4\times t_{SYS}+[0\sim(1.5\times t_{Curr.}+0.5\times t_{Tar.})]$, where $t_{Curr.}$ indicates the current clock period, $t_{Tar.}$ indicates the target clock period and the t_{SYS} indicates the current system clock period.

Operating Mode Switching

The device can switch between operating modes dynamically allowing the user to select the best performance/power ratio for the present task in hand. In this way microcontroller operations that do not require high performance can be executed using slower clocks thus requiring less operating current and prolonging battery life in portable applications.

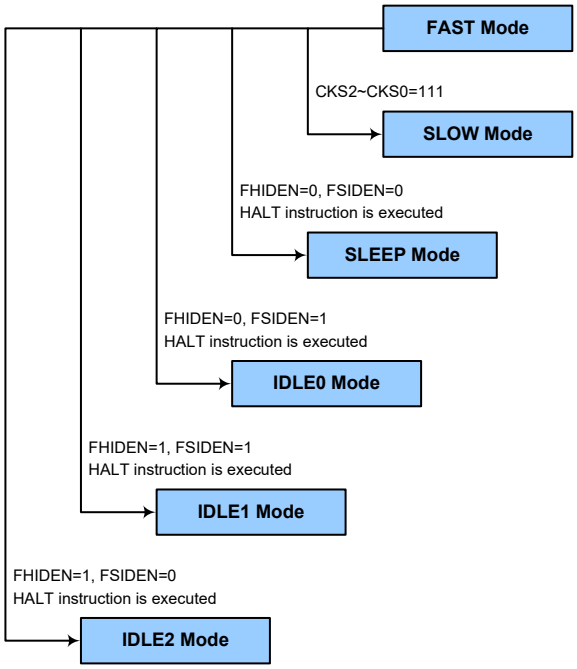
In simple terms, mode switching between the FAST Mode and SLOW Mode is executed using the CKS2~CKS0 bits in the SCC register while mode switching from the FAST/SLOW Modes to the SLEEP/IDLE Modes is executed via the HALT instruction. When an HALT instruction is executed, whether the device enters the IDLE Mode or the SLEEP Mode is determined by the condition of the FHIDEN and FSIDEN bits in the SCC register.



FAST Mode to SLOW Mode Switching

When running in the FAST Mode, which uses the high speed system oscillator, and therefore consumes more power, the system clock can switch to run in the SLOW Mode by set the CKS2~CKS0 bits to “111” in the SCC register. This will then use the low speed system oscillator which will consume less power. Users may decide to do this for certain operations which do not require high performance and can subsequently reduce power consumption.

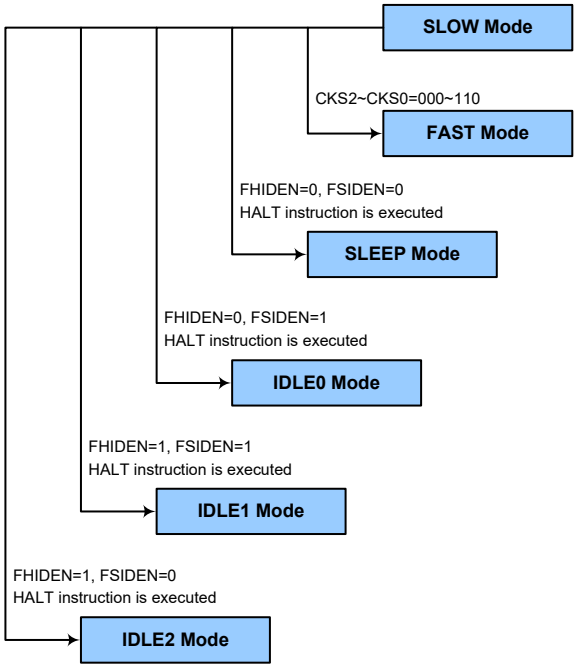
The SLOW Mode system clock is sourced from the LIRC oscillator and therefore requires this oscillator to be stable before full mode switching occurs.



SLOW Mode to FAST Mode Switching

In the SLOW mode the system clock is derived from f_{SUB}. When system clock is switched back to the FAST mode from f_{SUB}, the CKS2~CKS0 bits should be set to "000"~"110" and then the system clock will respectively be switched to f_H~f_H/64.

However, if f_H is not used in the SLOW mode and thus switched off, it will take some time to re-oscillate and stabilise when switching to the FAST mode from the SLOW Mode. This is monitored using the HIRCF bit in the SCC register. The time duration required for the high speed system oscillator stabilisation is specified in the System Start Up Time Characteristics.



Entering the SLEEP Mode

There is only one way for the device to enter the SLEEP Mode and that is to execute the "HALT" instruction in the application program with both the FHIDEN and FSIDEN bits in the SCC register equal to "0". In this mode all the clocks and functions will be switched off except the WDT function. When this instruction is executed under the conditions described above, the following will occur:

- The system clock will be stopped and the application program will stop at the "HALT" instruction.
- The Data Memory contents and registers will maintain their present condition.
- The I/O ports will maintain their present conditions.
- In the status register, the Power Down flag, PDF, will be set and the Watchdog time-out flag, TO, will be cleared.
- The WDT will be cleared and resume counting if the WDT function is enabled. If the WDT function is disabled, the WDT will be cleared and then stopped.

Entering the IDLE0 Mode

There is only one way for the device to enter the IDLE0 Mode and that is to execute the "HALT" instruction in the application program with the FHIDEN bit in the SCC register equal to "0" and the FSIDEN bit in the SCC register equal to "1". When this instruction is executed under the conditions described above, the following will occur:

- The f_H clock will be stopped and the application program will stop at the "HALT" instruction, but the f_{SUB} clock will be on.
- The Data Memory contents and registers will maintain their present condition.
- The I/O ports will maintain their present conditions.
- In the status register, the Power Down flag, PDF, will be set and the Watchdog time-out flag, TO, will be cleared.
- The WDT will be cleared and resume counting if the WDT function is enabled. If the WDT function is disabled, the WDT will be cleared and then stopped.

Entering the IDLE1 Mode

There is only one way for the device to enter the IDLE1 Mode and that is to execute the "HALT" instruction in the application program with both the FHIDEN and FSIDEN bits in the SCC register equal to "1". When this instruction is executed under the conditions described above, the following will occur:

- The f_H and f_{SUB} clocks will be on but the application program will stop at the "HALT" instruction.
- The Data Memory contents and registers will maintain their present condition.
- The I/O ports will maintain their present conditions.
- In the status register, the Power Down flag, PDF, will be set and the Watchdog time-out flag, TO, will be cleared.
- The WDT will be cleared and resume counting if the WDT function is enabled. If the WDT function is disabled, the WDT will be cleared and then stopped.

Entering the IDLE2 Mode

There is only one way for the device to enter the IDLE2 Mode and that is to execute the "HALT" instruction in the application program with the FHIDEN bit in the SCC register equal to "1" and the FSIDEN bit in the SCC register equal to "0". When this instruction is executed under the conditions described above, the following will occur:

- The f_H clock will be on but the f_{SUB} clock will be off and the application program will stop at the "HALT" instruction.

- The Data Memory contents and registers will maintain their present condition.
- The I/O ports will maintain their present conditions.
- In the status register, the Power Down flag, PDF, will be set and the Watchdog time-out flag, TO, will be cleared.
- The WDT will be cleared and resume counting if the WDT function is enabled. If the WDT function is disabled, the WDT will be cleared and then stopped.

Standby Current Considerations

As the main reason for entering the SLEEP or IDLE Mode is to keep the current consumption of the device to as low a value as possible, perhaps only in the order of several micro-amps except in the IDLE1 and IDLE2 Mode, there are other considerations which must also be taken into account by the circuit designer if the power consumption is to be minimised. Special attention must be made to the I/O pins on the device. All high-impedance input pins must be connected to either a fixed high or low level as any floating input pins could create internal oscillations and result in increased current consumption. This also applies to the device which has different package types, as there may be unbonded pins. These must either be set as outputs or if set as inputs must have pull-high resistors connected. In addition, the I/O pin-shared with VPP must not be set to output high, as this could result in increased current consumption.

Care must also be taken with the loads, which are connected to I/O pins, which are set as outputs. These should be placed in a condition in which minimum current is drawn or connected only to external circuits that do not draw current, such as other CMOS inputs. Also note that additional standby current will also be required if the LIRC oscillator has enabled.

In the IDLE1 and IDLE2 Mode the high speed oscillator is on, if the peripheral function clock source is derived from the high speed oscillator, the additional standby current will also be perhaps in the order of several hundred micro-amps.

Wake-up

To minimise power consumption the device can enter the SLEEP or any IDLE Mode, where the CPU will be switched off. However, when the device is woken up again, it will take a considerable time for the original system oscillator to restart, stabilise and allow normal operation to resume.

After the system enters the SLEEP or IDLE Mode, it can be woken up from one of various sources listed as follows:

- An external falling edge on Port A
- An external $\overline{\text{RES}}$ pin reset
- A system interrupt
- A WDT overflow

When the device executes the “HALT” instruction, it will enter the IDLE or SLEEP mode and the PDF flag will be set to 1. The PDF flag will be cleared to 0 if the device experiences a system power-up or executes the clear Watchdog Timer instruction.

If the system is woken up by an external $\overline{\text{RES}}$ pin reset, the device will experience a full system reset, however, if the device is woken up by a WDT overflow, a Watchdog Timer reset will be initiated. Although both of these wake-up methods will initiate a reset operation, the actual source of the wake-up can be determined by examining the TO and PDF flag. The PDF flag is cleared by a system power-up or executing the clear Watchdog Timer instructions and is set when executing the “HALT” instruction. The TO flag is set if a WDT time-out occurs and causes a wake-up that only resets the Program Counter and Stack Pointer, other flags remain in their original status.

Each pin on Port A can be set using the PAWU register to permit a negative transition on the pin to wake-up the system. When a pin wake-up occurs, the program will resume execution at the instruction following the "HALT" instruction. If the system is woken up by an interrupt, then two possible situations may occur. The first is where the related interrupt is disabled or the interrupt is enabled but the stack is full, in which case the program will resume execution at the instruction following the "HALT" instruction. In this situation, the interrupt which woke-up the device will not be immediately serviced, but will rather be serviced later when the related interrupt is finally enabled or when a stack level becomes free. The other situation is where the related interrupt is enabled and the stack is not full, in which case the regular interrupt response takes place. If an interrupt request flag is set high before entering the SLEEP or IDLE Mode, the wake-up function of the related interrupt will be disabled.

Watchdog Timer

The Watchdog Timer is provided to prevent program malfunctions or sequences from jumping to unknown locations, due to certain uncontrollable external events such as electrical noise.

Watchdog Timer Clock Source

The Watchdog Timer clock source is provided by the internal clock, f_{LIRC} which is sourced from the LIRC oscillator. The LIRC internal oscillator has an approximate frequency of 32kHz and this specified internal clock period can vary with V_{DD}, temperature and process variations. The Watchdog Timer source clock is then subdivided by a ratio of $[(2^8\text{-}2^0)\sim2^8]\sim[(2^{15}\text{-}2^7)\sim2^{15}]$ to give longer timeouts, the actual value being chosen using the WS2~WS0 bits in the WDTC register.

Watchdog Timer Control Register

A single register, WDTC, controls the required time-out period as well as the enable/disable operation.

- **WDTC Register**

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	WDTEN	WS2	WS1	WS0
R/W	—	—	—	—	R/W	R/W	R/W	R/W
POR	—	—	—	—	1	1	1	1

Bit 7~4 Unimplemented, read as "0"

Bit 3 **WDTEN**: WDT function enable control
0: Disable
1: Enable

Bit 2~0 **WS2~WS0**: WDT time-out period selection
000: $[(2^8\text{-}2^0)\sim2^8]/f_{LIRC}$
001: $[(2^9\text{-}2^1)\sim2^9]/f_{LIRC}$
010: $[(2^{10}\text{-}2^2)\sim2^{10}]/f_{LIRC}$
011: $[(2^{11}\text{-}2^3)\sim2^{11}]/f_{LIRC}$
100: $[(2^{12}\text{-}2^4)\sim2^{12}]/f_{LIRC}$
101: $[(2^{13}\text{-}2^5)\sim2^{13}]/f_{LIRC}$
110: $[(2^{14}\text{-}2^6)\sim2^{14}]/f_{LIRC}$
111: $[(2^{15}\text{-}2^7)\sim2^{15}]/f_{LIRC}$
These three bits determine the division ratio of the Watchdog Timer source clock, which in turn determines the timeout period.

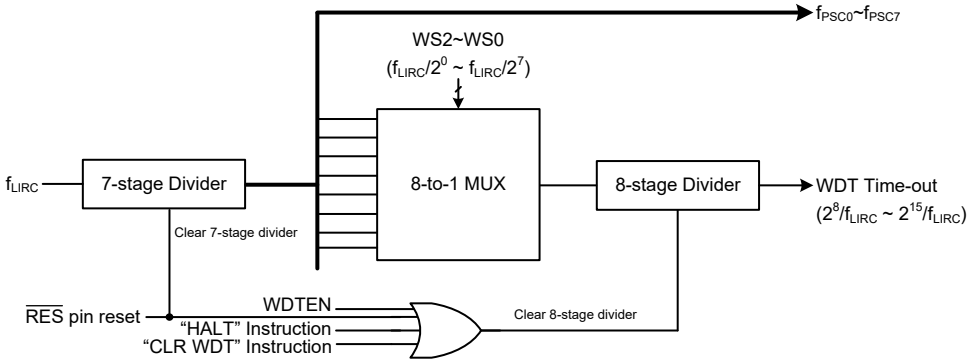
Watchdog Timer Operation

The Watchdog Timer operates by providing a device reset when its timer overflows. This means that in the application program and during normal operation the user has to strategically clear the Watchdog Timer before it overflows to prevent the Watchdog Timer from executing a reset. This is done using the clear watchdog instruction. If the program malfunctions for whatever reason, jumps to an unknown location, or enters an endless loop, the clear instruction will not be executed in the correct manner, in which case the Watchdog Timer will overflow and reset the device. With regard to the Watchdog Timer enable/disable function, there is one bit, WDTEN, in the WDTC register to offer the enable/disable control.

Under normal program operation, a Watchdog Timer time-out will initialise a device reset and set the status bit TO. However, if the system is in the SLEEP or IDLE Mode, when a Watchdog Timer time-out occurs, the TO bit in the status register will be set and only the Program Counter and Stack Pointer will be reset. Four methods can be adopted to clear the contents of the Watchdog Timer. The first is using the Watchdog Timer software clear instruction, the second is via a HALT instruction. The third is an external hardware reset, which means a low level on the external reset pin. The fourth is that the WDTEN bit is cleared to zero to disable the Watchdog Timer.

There is only one method of using software instruction to clear the Watchdog Timer. That is to use the single "CLR WDT" instruction to clear the WDT.

The maximum time-out period is when the 2^{15} division ratio is selected. As an example, with a 32kHz LIRC oscillator as its source clock, this will give a maximum watchdog period of around 1 second for the 2^{15} division ratio, and a minimum timeout of 8ms for the 2^8 division ration.



Watchdog Timer

Reset and Initialisation

A reset function is a fundamental part of any microcontroller ensuring that the device can be set to some predetermined condition irrespective of outside parameters. The most important reset condition is after power is first applied to the microcontrollers. In this case, internal circuitry will ensure that the microcontrollers, after a short delay, will be in a well-defined state and ready to execute the first program instruction. After this power-on reset, certain important internal registers will be set to defined states before the program commences. One of these registers is the Program Counter, which will be reset to zero forcing the microcontroller to begin program execution from the lowest Program Memory address.

In addition to the power-on reset, situations may arise where it is necessary to forcefully apply a reset condition when the device is running. One example of this is where after power has been applied and the device is already running, the $\overline{RES}$ line is forcefully pulled low. In such a case, known as a normal operation reset, some of the registers remain unchanged allowing the device to proceed with normal operation after the reset line is allowed to return high.

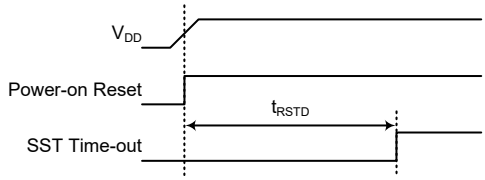
Another reset exists in the form of a Low Voltage Reset, LVR, where a full reset, similar to the $\overline{RES}$ reset is implemented in situations where the power supply voltage falls below a certain threshold. Another type of reset is when the Watchdog Timer overflows and resets the microcontroller. All types of reset operations result in different register conditions being setup.

Reset Functions

There are several ways in which a microcontroller reset can occur, through events occurring both internally and externally.

Power-on Reset

The most fundamental and unavoidable reset is the one that occurs after power is first applied to the microcontrollers. As well as ensuring that the Program Memory begins execution from the first memory address, a power-on reset also ensures that certain other registers are preset to known conditions. All the I/O port and port control registers will power up in a high condition ensuring that all pins will be first set to inputs.



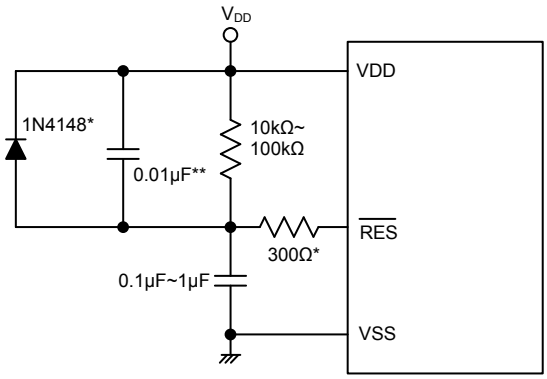
Power-on Reset Timing Chart

$\overline{RES}$ Pin Reset

As the reset pin is shared with an I/O pin, the reset function must be selected using a configuration option. Although the microcontroller has an internal RC reset function, if the V_{DD} power supply rise time is not fast enough or does not stabilise quickly at power-on, the internal reset function may be incapable of providing proper reset operation. For this reason it is recommended that an external RC network is connected to the $\overline{RES}$ pin, whose additional time delay will ensure that the $\overline{RES}$ pin remains low for an extended period to allow the power supply to stabilise. During this time delay, normal operation of the microcontroller will be inhibited. After the $\overline{RES}$ line reaches a certain voltage value, the reset delay time t_{RSTD} is invoked to provide an extra delay time after which the microcontroller will begin normal operation. The abbreviation SST in the figures stands for System Start-up Timer.

For most applications a resistor connected between VDD and the $\overline{RES}$ pin and a capacitor connected between VSS and the $\overline{RES}$ pin will provide a suitable external reset circuit. Any wiring connected to the $\overline{RES}$ pin should be kept as short as possible to minimise any stray noise interference.

For applications that operate within an environment where more noise is present the Enhanced Reset Circuit shown is recommended.

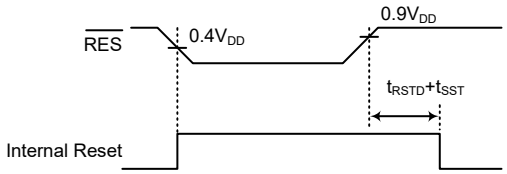


Note: * It is recommended that this component is added for added ESD protection.

** It is recommended that this component is added in environments where power line noise is significant.

External $\overline{\text{RES}}$ Circuit

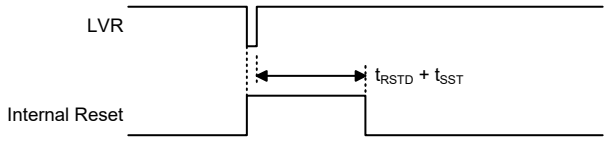
Pulling the $\overline{\text{RES}}$ pin low using external hardware will also execute a device reset. In this case, as in the case of other resets, the Program Counter will reset to zero and program execution initiated from this point.



$\overline{\text{RES}}$ Reset Timing Chart

Low Voltage Reset – LVR

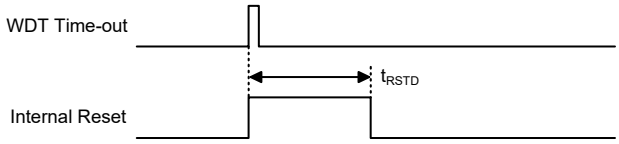
The microcontroller contains a low voltage reset circuit in order to monitor the supply voltage of the device. The LVR function can be enabled or disabled which is selected by the configuration options. If the supply voltage of the device drops to within a range of 0.9V~V_{LVR} such as might occur when changing the battery in battery powered applications, the LVR will automatically reset the device internally. For a valid LVR signal, a low supply voltage, i.e., a voltage in the range between 0.9V~V_{LVR} must exist for a time greater than that specified by t_{LVR} in the LVR Electrical characteristics. If the low supply voltage state does not exceed this value, the LVR will ignore the low supply voltage and will not perform a reset function. The specified voltage values for V_{LVR} can be selected using configuration options.



Low Voltage Reset Timing Chart

Watchdog Time-out Reset during Normal Operation

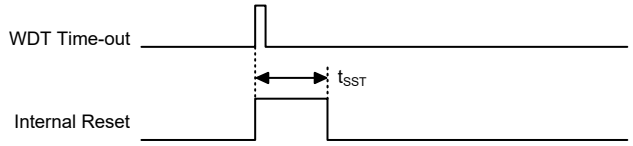
When the Watchdog time-out Reset during normal operations in the FAST or SLOW mode occurs, the Watchdog time-out flag TO will be set to "1".



WDT Time-out Reset during Normal Operation Timing Chart

Watchdog Time-out Reset during SLEEP or IDLE Mode

The Watchdog time-out Reset during SLEEP or IDLE Mode is a little different from other kinds of reset. Most of the conditions remain unchanged except that the Program Counter and the Stack Pointer will be cleared to "0" and the TO and PDF flags will be set to "1". Refer to the System Start Up Time Characteristics for t_{SST} details.



WDT Time-out Reset during SLEEP or IDLE Mode Timing Chart

Reset Initial Conditions

The different types of reset described affect the reset flags in different ways. These flags, known as PDF and TO are located in the status register and are controlled by various microcontroller operations, such as the SLEEP or IDLE Mode function or Watchdog Timer. The reset flags are shown in the table:

TO	PDF	Reset Conditions
0	0	Power-on reset
u	u	$\overline{RES}$ or LVR reset during FAST or SLOW Mode operation
1	u	WDT time-out reset during FAST or SLOW Mode operation
1	1	WDT time-out reset during IDLE or SLEEP Mode operation

"u" stands for unchanged

The following table indicates the way in which the various components of the microcontroller are affected after a power-on reset occurs.

Item	Condition After Reset
Program Counter	Reset to zero
Interrupts	All interrupts will be disabled
WDT, Time Base	Cleared after reset, WDT begins counting
Timer/Event Counter	Timer/Event Counter will be turned off
Input/Output Ports	I/O ports will be set as inputs
Stack Pointer	Stack Pointer will point to the top of the stack

The different kinds of resets all affect the internal registers of the microcontroller in different ways. To ensure reliable continuation of normal program execution after a reset occurs, it is important to know what condition the microcontroller is in after a particular reset occurs. The following table describes how each type of reset affects each of the microcontroller internal registers.

Register	Reset (Power On)	WDT Time-out (Normal Operation)	$\overline{\text{RES}}$ Reset (Normal Operation)	WDT Time-out (IDLE/SLEEP)
IAR0	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu
MP0	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu
IAR1	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu
MP1	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu
ACC	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu
PCL	0000 0000	0000 0000	0000 0000	0000 0000
TBLP	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu
TBLH	-xxx xxxx	-uuu uuuu	-uuu uuuu	-uuu uuuu
STATUS	--00 xxxx	--1u uuuu	--uu uuuu	--11 uuuu
INTEG	---- 0000	---- 0000	---- 0000	---- uuuu
WDTC	---- 1111	---- 1111	---- 1111	---- uuuu
TBC	0-00 -000	0-00 -000	0-00 -000	u-uu -uuu
SCC	111- 0000	111- 0000	111- 0000	uuu- uuuu
PAS0	0000 0000	0000 0000	0000 0000	uuuu uuuu
PAS1	-0-0 00-0	-0-0 00-0	-0-0 00-0	-u-u uu-u
PA	1111 1111	1111 1111	1111 1111	uuuu uuuu
PAC	1111 1111	1111 1111	1111 1111	uuuu uuuu
PAPU	0000 0000	0000 0000	0000 0000	uuuu uuuu
PAWU	0000 0000	0000 0000	0000 0000	uuuu uuuu
INTC0	-000 0000	-000 0000	-000 0000	-uuu uuuu
INTC1	-000 -000	-000 -000	-000 -000	-uuu- -uuu
TMR1C	00-0 1000	00-0 1000	00-0 1000	uu-u uuuu
TMR1L	0000 0000	0000 0000	0000 0000	uuuu uuuu
TMR1H	0000 0000	0000 0000	0000 0000	uuuu uuuu
IFS	0000 0000	0000 0000	0000 0000	uuuu uuuu
SADC0	000- 0000	000- 0000	000- 0000	uuu- uuuu
SADC1	---0 0000	---0 0000	---0 0000	---u uuuu
SADO	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu
PB	--11 1111	--11 1111	--11 1111	--uu uuuu
PBC	--11 1111	--11 1111	--11 1111	--uu uuuu
PBPU	--00 0000	--00 0000	--00 0000	--uu uuuu
PBS0	-000 0000	-000 0000	-000 0000	-uuu uuuu
PC	---- 1111	---- 1111	---- 1111	---- uuuu
PCC	---- 1111	---- 1111	---- 1111	---- uuuu
PCPU	---- 0000	---- 0000	---- 0000	---- uuuu
PCS0	---- 0000	---- 0000	---- 0000	---- uuuu
SCOMC	0000 ----	0000 ----	0000 ----	uuuu ----
PBS1	---- -0-0	---- -0-0	---- -0-0	---- -u-u
OCR	---- 00--	---- 00--	---- 00--	---- uu--
ODL	0000 0000	0000 0000	0000 0000	uuuu uuuu
ODH	-000 0000	-000 0000	-000 0000	-uuu uuuu
TMR0C/PWMC	00-0 0000	00-0 0000	00-0 0000	uu-u uuuu
TMR0/PWMADATA	0000 0000	0000 0000	0000 0000	uuuu uuuu

Register	Reset (Power On)	WDT Time-out (Normal Operation)	$\overline{RES}$ Reset (Normal Operation)	WDT Time-out (IDLE/SLEEP)
PWMBDATA	0000 0000	0000 0000	0000 0000	uuuu uuuu
PWMCDATA	0000 0000	0000 0000	0000 0000	uuuu uuuu

Note: "u" stands for unchanged
"x" stands for unknown
"-" stands for unimplemented

Input/Output Ports

The device offers considerable flexibility on their I/O ports. With the input or output designation of every pin fully under user program control, pull-high selections for all ports and wake-up selections on certain pins, the user is provided with an I/O structure to meet the needs of a wide range of application possibilities.

The device provides bidirectional input/output lines labeled with port names PA~PC. These I/O ports are mapped to the RAM Data Memory with specific addresses as shown in the Special Purpose Data Memory table. All of these I/O ports can be used for input and output operations. For input operation, these ports are non-latching, which means the inputs must be ready at the T2 rising edge of instruction "MOV A, [m]", where m denotes the port address. For output operation, all the data is latched and remains unchanged until the output latch is rewritten.

Register Name	Bit							
	7	6	5	4	3	2	1	0
PA	PA7	PA6	PA5	PA4	PA3	PA2	PA1	PA0
PAC	PAC7	PAC6	PAC5	PAC4	PAC3	PAC2	PAC1	PAC0
PAPU	PAPU7	PAPU6	PAPU5	PAPU4	PAPU3	PAPU2	PAPU1	PAPU0
PAWU	PAWU7	PAWU6	PAWU5	PAWU4	PAWU3	PAWU2	PAWU1	PAWU0
PB	—	—	PB5	PB4	PB3	PB2	PB1	PB0
PBC	—	—	PBC5	PBC4	PBC3	PBC2	PBC1	PBC0
PBPU	—	—	PBPU5	PBPU4	PBPU3	PBPU2	PBPU1	PBPU0
PC	—	—	—	—	PC3	PC2	PC1	PC0
PCC	—	—	—	—	PCC3	PCC2	PCC1	PCC0
PCPU	—	—	—	—	PCPU3	PCPU2	PCPU1	PCPU0

"—": Unimplemented, read as "0"

I/O Logic Function Register List

Pull-high Resistors

Many product applications require pull-high resistors for their switch inputs usually requiring the use of an external resistor. To eliminate the need for these external resistors, all I/O pins, when configured as a digital input have the capability of being connected to an internal pull-high resistor. These pull-high resistors are selected using the relevant pull-high control registers and are implemented using weak PMOS transistors.

Note that the pull-high resistor can be controlled by the relevant pull-high control register only when the pin-shared functional pin is selected as a digital input. Otherwise, the pull-high resistors cannot be enabled.

• **PxPU Register**

Bit	7	6	5	4	3	2	1	0
Name	PxPU7	PxPU6	PxPU5	PxPU4	PxPU3	PxPU2	PxPU1	PxPU0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	0	0	0	0	0

PxPUn: I/O Port x Pin pull-high function control
0: Disable
1: Enable
The PxPUn bit is used to control the pin pull-high function. Here the "x" can be A, B, or C. However, the actual available bits for each I/O Port may be different.

Port A Wake-up

The HALT instruction forces the microcontroller into the SLEEP or IDLE Mode which preserves power, a feature that is important for battery and other low-power applications. Various methods exist to wake-up the microcontroller, one of which is to change the logic condition on one of the Port A pins from high to low. This function is especially suitable for applications that can be woken up via external switches. Each pin on Port A can be selected individually to have this wake-up feature using the PAWU register.

Note that the wake-up function can be controlled by the wake-up control registers only when the pin is selected as a general purpose input and the MCU enters the IDLE or SLEEP mode.

• **PAWU Register**

Bit	7	6	5	4	3	2	1	0
Name	PAWU7	PAWU6	PAWU5	PAWU4	PAWU3	PAWU2	PAWU1	PAWU0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	0	0	0	0	0

PAWUn: Port A Pin wake-up function control
0: Disable
1: Enable

I/O Port Control Registers

Each I/O port has its own control register known as PAC~PCC, to control the input/output configuration. With this control register, each CMOS output or input can be reconfigured dynamically under software control. Each pin of the I/O ports is directly mapped to a bit in its associated port control register. For the I/O pin to function as an input, the corresponding bit of the control register must be written as a "1". This will then allow the logic state of the input pin to be directly read by instructions. When the corresponding bit of the control register is written as a "0", the I/O pin will be set as a CMOS output. If the pin is currently set as an output, instructions can still be used to read the output register. However, it should be noted that the program will in fact only read the status of the output data latch and not the actual logic status of the output pin.

• **PxC Register**

Bit	7	6	5	4	3	2	1	0
Name	PxC7	PxC6	PxC5	PxC4	PxC3	PxC2	PxC1	PxC0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	1	1	1	1	1	1	1	1

PxCn: I/O Port A Pin type selection
0: Output
1: Input

The PxCn bit is used to control the pin type selection. Here the "x" can be A, B, or C. However, the actual available bits for each I/O Port may be different.

Pin-shared Functions

The flexibility of the microcontroller range is greatly enhanced by the use of pins that have more than one function. Limited numbers of pins can force serious design constraints on designers but by supplying pins with multi-functions, many of these difficulties can be overcome. For these pins, the desired function of the multi-function I/O pins is selected by a series of registers via the application program control.

Pin-shared Function Selection Registers

The limited number of supplied pins in a package can impose restrictions on the amount of functions a certain device can contain. However by allowing the same pins to share several different functions and providing a means of function selection, a wide range of different functions can be incorporated into even relatively small package sizes. The device includes a Port "x" output function selection register "n", labeled as PxSn, and Input Function Selection register, labeled as IFS, which can select the desired functions of the multi-function pin-shared pins.

The most important point to note is to make sure that the desired pin-shared function is properly selected and also deselected. For most pin-shared functions, to select the desired pin-shared function, the pin-shared function should first be correctly selected using the corresponding pin-shared control register. After that the corresponding peripheral functional setting should be configured and then the peripheral function can be enabled. However, a special point must be noted for the digital input pin TCn, etc, which shares the same pin-shared control configuration with its corresponding general purpose I/O function when setting the relevant pin-shared control bit. To select this pin function, in addition to the necessary pin-shared control and peripheral functional setup aforementioned, it must also be set as an input by setting the corresponding bit in the I/O port control register. To correctly deselect the pin-shared function, the peripheral function should first be disabled and then the corresponding pin-shared function control register can be modified to select other pin-shared functions.

Register Name	Bit							
	7	6	5	4	3	2	1	0
PAS0	PAS07	PAS06	PAS05	PAS04	PAS03	PAS02	PAS01	PAS00
PAS1	—	PAS16	—	PAS14	PAS13	PAS12	—	PAS10
PBS0	—	PBS06	PBS05	PBS04	PBS03	PBS02	PBS01	PBS00
PBS1	—	—	—	—	—	PBS12	—	PBS10
PCS0	—	—	—	—	PCS03	PCS02	PCS01	PCS00
IFS	TC1PS1	TC1PS0	TC0PS1	TC0PS0	INT1PS1	INT1PS0	INT0PS1	INT0PS0

Pin-shared Function Selection Register List

- **PAS0 Register**

Bit	7	6	5	4	3	2	1	0
Name	PAS07	PAS06	PAS05	PAS04	PAS03	PAS02	PAS01	PAS00
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	0	0	0	0	0

Bit 7~6 **PAS07~PAS06**: PA3 Pin-Shared function selection
00: PA3/INT0
01: SCOM5
10: AN0
11: PA3/INT0

Bit 5~4 **PAS05~PAS04**: PA2 Pin-Shared function selection
00: PA2/TC0
01: SCOM6
10: AN1
11: PA2/TC0

Bit 3~2 **PAS03~PAS02**: PA1 Pin-Shared function selection
00: PA1
01: PWMAOB
10: VREF
11: AN2

Bit 1~0 **PAS01~PAS00**: PA0 Pin-Shared function selection
00: PA0/INT1
01: PWMAO
10: AN3
11: PA0/INT1

• PAS1 Register

Bit	7	6	5	4	3	2	1	0
Name	—	PAS16	—	PAS14	PAS13	PAS12	—	PAS10
R/W	—	R/W	—	R/W	R/W	R/W	—	R/W
POR	—	0	—	0	0	0	—	0

Bit 7 Unimplemented, read as "0"

Bit 6 **PAS16**: PA7 Pin-Shared function selection
0: PA7/$\overline{\text{RES}}$/VPP
1: PWMAO

Bit 5 Unimplemented, read as "0"

Bit 4 **PAS14**: PA6 Pin-Shared function selection
0: PA6/TC1/INT1
1: PWMCO

Bit 3~2 **PAS13~PAS12**: PA5 Pin-Shared function selection
00: PA5
01: PWMBO
10: AN9
11: PA5

Bit 1 Unimplemented, read as "0"

Bit 0 **PAS10**: PA4 Pin-Shared function selection
0: PA4/TC1
1: PWMAO

• PBS0 Register

Bit	7	6	5	4	3	2	1	0
Name	—	PBS06	PBS05	PBS04	PBS03	PBS02	PBS01	PBS00
R/W	—	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	—	0	0	0	0	0	0	0

Bit 7 Unimplemented, read as "0"

Bit 6 **PBS06**: PB3 Pin-Shared function selection
0: PB3/INT0
1: SCOM3

Bit 5~4 **PBS05~PBS04**: PB2 Pin-Shared function selection
00: PB2/INT0
01: SCOM2
10: AN8
11: PB2/INT0

Bit 3~2 **PBS03~PBS02**: PB1 Pin-Shared function selection
00: PB1/INT1
01: SCOM1
10: AN7
11: PWMCO

Bit 1~0 **PBS01~PBS00**: PB0 Pin-Shared function selection
00: PB0/TC0
01: SCOM0
10: AN6
11: PWMBO

• PBS1 Register

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	—	PBS12	—	PBS10
R/W	—	—	—	—	—	R/W	—	R/W
POR	—	—	—	—	—	0	—	0

Bit 7~3 Unimplemented, read as "0"

Bit 2 **PBS12**: PB5 Pin-Shared function selection
0: PB5
1: PWMAO

Bit 1 Unimplemented, read as "0"

Bit 0 **PBS10**: PB4 Pin-Shared function selection
0: PB4/TC1/INT1
1: SCOM4

• PCS0 Register

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	PCS03	PCS02	PCS01	PCS00
R/W	—	—	—	—	R/W	R/W	R/W	R/W
POR	—	—	—	—	0	0	0	0

Bit 7~4 Unimplemented, read as "0"

Bit 3 **PCS03**: PC2 Pin-Shared function selection
0: PC2/TC0
1: PWMAO

Bit 2 **PCS02**: PC1 Pin-Shared function selection
0: PC1/INT0/TC1
1: AN5

Bit 1~0 **PCS01~PCS00**: PC0 Pin-Shared function selection
00: PC0
01: AN4
10: VREF
11: PC0

• **IFS Register**

Bit	7	6	5	4	3	2	1	0
Name	TC1PS1	TC1PS0	TC0PS1	TC0PS0	INT1PS1	INT1PS0	INT0PS1	INT0PS0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	0	0	0	0	0

Bit 7~6 **TC1PS1~TC1PS0**: TC1 input source pin selection
00: PA4
01: PA6
10: PC1
11: PB4

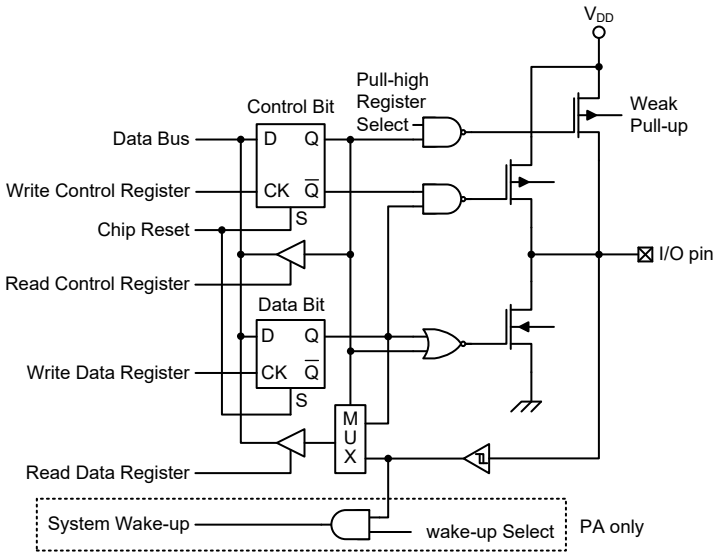
Bit 5~4 **TC0PS1~TC0PS0**: TC0 input source pin selection
00: PA2
01: PB0
10: PC2
11: PC3

Bit 3~2 **INT1PS1~INT1PS0**: INT1 input source pin selection
00: PB1
01: PA0
10: PB4
11: PA6

Bit 1~0 **INT0PS1~INT0PS0**: INT0 input source pin selection
00: PB2
01: PA3
10: PC1
11: PB3

I/O Pin Structure

The accompanying diagram illustrates the internal structure of the I/O logic function. As the exact logical construction of the I/O pin will differ from this drawing, it is supplied as a guide only to assist with the functional understanding of the I/O logic function. The wide range of pin-shared structures does not permit all types to be shown.



Logic Function Input/Output Structure

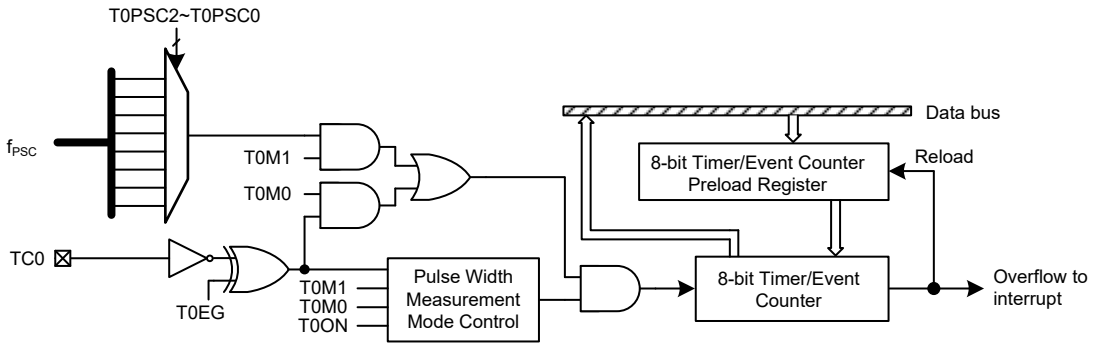
Programming Considerations

Within the user program, one of the first things to consider is port initialisation. After a reset, all of the I/O data and port control registers will be set high. This means that all I/O pins will default to an input state, the level of which depends on the other connected circuitry and whether pull-high selections have been chosen. If the port control registers are then programmed to set some pins as outputs, these output pins will have an initial high output value unless the associated port data registers are first programmed. Selecting which pins are inputs and which are outputs can be achieved byte-wide by loading the correct values into the appropriate port control register or by programming individual bits in the port control register using the "SET [m].i" and "CLR [m].i" instructions. Note that when using these bit control instructions, a read-modify-write operation takes place. The microcontroller must first read in the data on the entire port, modify it to the required new bit values and then rewrite this data back to the output ports.

Port A has the additional capability of providing wake-up function. When the device is in the SLEEP or IDLE Mode, various methods are available to wake the device up. One of these is a high to low transition of any of the Port A pins. Single or multiple pins on Port A can be set to have this function.

8-bit Timer/Event Counter

The provision of the Timer/Event Counter forms an important part of any microcontroller, giving the designer a means of carrying out time related functions. The device contains an 8-bit Timer/Event Counter, which is an 8-bit programmable count-up counter and the clock may come from an external or internal clock source. As this timer has four different operating modes, it can be configured to operate as a general timer, an external event counter, a pulse width measurement or a pulse width mode.



8-bit Timer/Event Counter Structure

Timer/Event Counter Input Clock Source

The Timer/Event Counter clock source can originate from various sources, an internal clock or an external pin. The internal clock source is used when the timer is in the Timer Mode, Pulse Width Measurement Mode and PWM Mode. For the Timer/Event Counter, this internal clock is sourced from the Time Base prescaler, which is selected by the T0PSC2~T0PSC0 bits in the TMR0C Timer/Event Control Register.

An external clock source is used when the Timer/Event Counter is in the Event Counter Mode, the clock source is provided on the external TC0 pin. Depending upon the condition of the T0EG bit, each high to low or low to high transition on the external timer pin will increase the counter by one.

Timer/Event Counter Registers

There are several registers related to the Timer/Event Counter. The first is the TMR0 register that contains the actual value of the timer and into which an initial value can be preloaded. Writing to the TMR0 register will transfer the specified data to the Timer/Event Counter. Reading the TMR0 register will read the contents of the Timer/Event Counter. The second is the TMR0C control register, which is used to define the operating mode, control the counting enable or disable and select the active edge. Another four registers control the overall operation of the Pulse Width Modulator, the data register, PWMADATA, PWMBDATA, PWMCDATA and the control register, PWMC. Note that the PWMC and TMR0C registers have the same register address, the PWMADATA and TMR0 registers also have the same register address.

Register Name	Bit							
	7	6	5	4	3	2	1	0
TMR0	D7	D6	D5	D4	D3	D2	D1	D0
TMR0C	T0M1	T0M0	—	T0ON	T0EG	T0PSC2	T0PSC1	T0PSC0
PWMC	T0M1	T0M0	—	PWMDIV2	PWMDIV1	PWMDIV0	PWMSEL	PWMEN
PWMADATA	PWMAD7	PWMAD6	PWMAD5	PWMAD4	PWMAD3	PWMAD2	PWMAD1	PWMAD0
PWMBDATA	PWMBD7	PWMBD6	PWMBD5	PWMBD4	PWMBD3	PWMBD2	PWMBD1	PWMBD0
PWMCDATA	PWMCD7	PWMCD6	PWMCD5	PWMCD4	PWMCD3	PWMCD2	PWMCD1	PWMCD0

Timer/Event Counter Register List

Timer Register – TMR0

The timer register TMR0 is the place where the actual timer value is stored. The value in the timer register increases by one each time an internal clock pulse is received or an external transition occurs on the external timer pin. The timer will count from the initial value loaded by the preload register to the full count of FFH for the 8-bit Timer/Event Counter, at which point the timer overflows and an internal interrupt signal is generated. The timer value will then be loaded with the preload register value and continue counting.

Note that to achieve a maximum full range count of FFH, the preload register must first be cleared. Note that if the Timer/Event Counter is in an off condition and data is written to its preload register, this data will be immediately written into the actual counter. However, if the counter is enabled and counting, any new data written into the preload data register during this period will remain in the preload register and will only be written into the actual counter until an overflow occurs.

- **TMR0 Register**

Bit	7	6	5	4	3	2	1	0
Name	D7	D6	D5	D4	D3	D2	D1	D0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	0	0	0	0	0

Bit 7~0 **D7~D0**: Timer preload register byte

Timer Control Register – TMR0C

The flexible features of the microcontroller Timer/Event Counter are implemented by operating in four different modes, the options of which are determined by the contents of control register bits.

The Timer Control Register is known as TMR0C. It is the Timer Control Register together with its corresponding timer register that controls the full operation of the Timer/Event Counter. Before the timer can be used, it is essential that the Timer Control Register is fully programmed with the right data to ensure its correct operation, a process that is normally carried out during program initialisation.

To select which of the four modes the timer is to operate in, namely the PWM Mode, Timer Mode, the Event Counter Mode or the Pulse Width Measurement Mode, the T0M1~T0M0 bits in the Timer Control Register must be set to the required logic levels. The timer-on bit T0ON provides the basic on/off control of the respective timer. Setting the bit to high allows the counter to run. Clearing the bit stops the counter. When the internal clock f_{PSC} is used, it can be selected by properly setting the T0PSC2~T0PSC0 bits. The internal clock selection will have no effect if an external clock source is used. If the timer is in the Event Counter or Pulse Width Measurement Mode, the active transition edge type is selected by the logic level of the T0EG bit in the TMR0C Register.

- **TMR0C Register**

Bit	7	6	5	4	3	2	1	0
Name	T0M1	T0M0	—	T0ON	T0EG	T0PSC2	T0PSC1	T0PSC0
R/W	R/W	R/W	—	R/W	R/W	R/W	R/W	R/W
POR	0	0	—	0	0	0	0	0

Bit 7~6 **T0M1~T0M0**: Timer/Event Counter operating mode selection
00: PWM Mode
01: Event Counter Mode
10: Timer Mode
11: Pulse Width Measurement Mode

Bit 5 Unimplemented, read as "0"

Bit 4 **T0ON**: Timer/Event Counter counting enable
0: Disable
1: Enable
Note that this bit is invalid in PWM Mode.

Bit 3 **T0EG**: Timer/Event Counter active edge selection
Event Counter Mode
0: Count on rising edge
1: Count on falling edge
Pulse Width Measurement Mode
0: Start counting on falling edge, stop on rising edge
1: Start counting on rising edge, stop on falling edge
PWM Mode
Unused

Bit 2~0 **T0PSC2~T0PSC0**: Timer internal clock selection
000: $f_{PSC}/2^0$
001: $f_{PSC}/2^1$
010: $f_{PSC}/2^2$
011: $f_{PSC}/2^3$
100: $f_{PSC}/2^4$
101: $f_{PSC}/2^5$
110: $f_{PSC}/2^6$
111: $f_{PSC}/2^7$

Timer PWM Mode Registers – PWMC, PWMADATA, PWMBDATA, PWMCDATA

There four registers control the overall operation of the Pulse Width Modulator channel. These are the data registers, PWMADATA, PWMBDATA, PWMCDATA and a single control register, PWMC. Note that the PWMC register is the same register address as TMR0C register. Additionally, the PWMADATA and TMR0 registers also have the same register address.

- **PWMC Register**

Bit	7	6	5	4	3	2	1	0
Name	T0M1	T0M0	—	PWMDIV2	PWMDIV1	PWMDIV0	PWMSEL	PWMEN
R/W	R/W	R/W	—	R/W	R/W	R/W	R/W	R/W
POR	0	0	—	0	0	0	0	0

Bit 7~6 **T0M1~T0M0**: Timer/Event Counter operating mode selection
00: PWM Mode
01: Event Counter Mode
10: Timer Mode
11: Pulse Width Measurement Mode

Bit 5 Unimplemented, read as "0"

Bit 4~2 **PWMDIV2~PWMDIV0**: f_{DIV} Frequency selection
000: $f_{DIV} = f_{SYS}$
001: $f_{DIV} = f_{SYS}/2$
010: $f_{DIV} = f_{SYS}/3$
011: $f_{DIV} = f_{SYS}/4$
100: $f_{DIV} = f_{SYS}/8$
101: $f_{DIV} = f_{SYS}/16$
110: $f_{DIV} = f_{SYS}/32$
111: $f_{DIV} = f_{SYS}/64$

Bit 1 **PWMSEL**: PWM mode selection
0: (6+2) bits mode
1: (7+1) bits mode

Bit 0 **PWMEN**: PWM enable control
0: Disable
1: Enable

Note: 1. When the PWMEN bit is cleared to zero to disable the PWM function, the internal PWMmO signal will be pulled low. However, the external PWMmO pin status will be floating when the multi-functional pin is selected as a PWMmO output and the PWMEN bit is cleared to zero. (m=A, B or C)
2. After the PWMEN bit is set high, the first PWM modulation cycle period and cycle may not match the expected waveform. The PWM output will be normal after the first PWM cycle.
3. The PWMAOB is the inverse output of the PWMAO, which can generate a complementary output.

- **PWMmDATA Register (m=A, B or C)**

Bit	7	6	5	4	3	2	1	0
Name	PWMmD7	PWMmD6	PWMmD5	PWMmD4	PWMmD3	PWMmD2	PWMmD1	PWMmD0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	0	0	0	0	0

Bit 7~0 **PWMmD7~PWMmD0**: PWM output duty cycle PWMmD bit 7 ~ bit 0

Note that the duty cycle value PWMmDATA, once being changed, will reflect on the PWMmO output signal immediately. Therefore, a sudden PWM duty change will occur in the current PWM cycle, resulting in undesired waveform, which only lasts a PWM cycle. Starting from the next new PWM cycle, the PWM duty will be in accordance with the new PWMmD value.

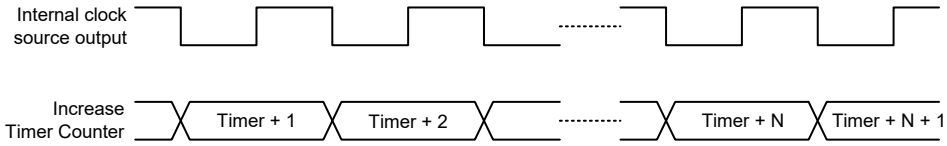
Timer/Event Counter Operating Modes

The Timer/Event Counter can operate in one of four operating modes, PWM Mode, Timer Mode, Event Counter Mode or Pulse Width Measurement Mode. The operating mode is selected using the T0M1 and T0M0 bits in the TMR0C register.

Timer Mode

To select this mode, bits T0M1 and T0M0 in the TMR0C register should be set to "10" respectively. In this mode, the Timer/Event Counter can be utilised to measure fixed time intervals, providing an internal interrupt signal each time the Timer/Event Counter overflows.

When operating in this mode the internal clock f_{PSC} is used as the timer clock. The clock source is from the Time Base prescaler, and is selected by the T0PSC2~T0PSC0 bits in the TMR0C register. The timer-on bit T0ON must be set high to enable the timer to run. Each time an internal clock high to low transition occurs, the timer increases by one. When the timer reaches its maximum 8-bit, FF Hex, value and overflows, an interrupt request is generated and thetimer will reload the value already loaded into the preload register and continue counting. It should be noted that in the Timer mode, even if the device is in the IDLE/SLEEP mode, if the selected internal clock is still activated the timer will continue to count.



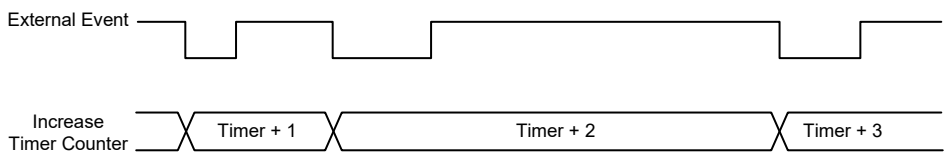
Timer Mode Timing Chart

Event Counter Mode

To select this mode, bits T0M1 and T0M0 in the TMR0C register should be set to "01" respectively. In this mode, a number of externally changing logic events, occurring on the external timer TC0 pin, can be recorded by the Timer/Event Counter.

When operating in this mode, the external timer pin, TC0, is used as the Timer/Event Counter clock source. After the other bits in the Timer Control Register have been properly configured, the enable bit T0ON, can be set high to enable the Timer/Event Counter. If the Active Edge Selection bit, T0EG, is low, the Timer/Event Counter will increase each time the TC0 pin receives a low to high transition. If the T0EG bit is high, the counter will increase each time the TC0 pin receives a high to low transition. When it is full and overflows, an interrupt signal is generated and the Timer/Event Counter will reload the value already loaded into the preload register and continue counting.

As the external timer pin TC0 is pin-shared with other pin functions, the TC0 pin function must first be selected using relevant pin-shared function selection bits. The pin must also be set as an input by setting the corresponding bit in the port control register. It should be noted that in the Event Counter mode, even if the device is in the IDLE/SLEEP Mode, the Timer/Event Counter will continue to record externally changing logic events on the TC0 pin. As a result when the timer overflows it will generate a timer interrupt and corresponding wake-up source.



Event Counter Mode Timing Chart (T0EG=1)

Pulse Width Measurement Mode

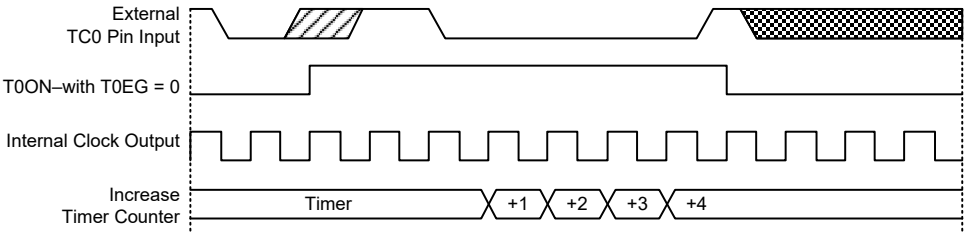
To select this mode, bits T0M1 and T0M0 in the TMR0C register should be set to "11" respectively. In this mode, the Timer/Event Counter can be utilised to measure the width of external pulses applied to the external timer pin.

When operating in this mode the internal clock f_{PSC} is used as the timer clock. The clock source is from the Time Base prescaler, the division ratio is determined by the T0PSC2~T0PSC0 bits in the TMR0C register. After the other bits in the Timer Control Register have been properly configured, the enable bit T0ON, can be set high to enable the Timer/Event Counter, however it will not actually start counting until an active edge is received on the TC0 pin.

If the active edge selection bit T0EG is low, once a high to low transition has been received on the TC0 pin, the Timer/Event Counter will start counting based on the internal selected clock source until the TC0 pin returns to its original high level. At this point the enable bit will be automatically reset to zero and the Timer/Event Counter will stop counting. If the Active Edge Selection bit is high, the Timer/Event Counter will begin counting once a low to high transition has been received on the TC0 pin and stop counting when the external timer pin returns to its original low level. As before, the enable bit will then be automatically reset to zero. It is important to note that in the pulse width measurement Mode, the enable bit is automatically reset to zero when the external control signal on the TC0 pin returns to its original level, whereas in the event count or timer mode the enable bit can only be reset to zero under application program control.

The residual value in the Timer/Event Counter, which can now be read by the program, therefore represents the length of the pulse received on the TC0 pin. As the enable bit has now been reset, any further transitions on the external timer pin will be ignored. The timer cannot begin further pulse width measurement until the enable bit is set high again by the application program. In this way, single shot pulse measurements can be easily made. It should be noted that in this mode the Timer/Event Counter is controlled by logical transitions on the external timer pin and not by the logic level. When the Timer/Event Counter is full and overflows, an interrupt signal is generated and the Timer/Event Counter will reload the value already loaded into the preload register and continue counting.

As the external timer pin TC0 is pin-shared with other pin functions, the TC0 pin function must first be selected using relevant pin-shared function selection bits. The pin must also be set as an input by setting the corresponding bit in the port control register. It should be noted that in the Pulse Width Capture mode, even if the device is in the IDLE/SLEEP Mode, the Timer/Event Counter will continue to record externally changing logic events on the TC0 pin if the internal clock source is still activated and the external signal continues to change state. As a result when the timer overflows it will generate a timer interrupt and corresponding wake-up source.



Internal Clock Output is sampled at every falling edge of T1.

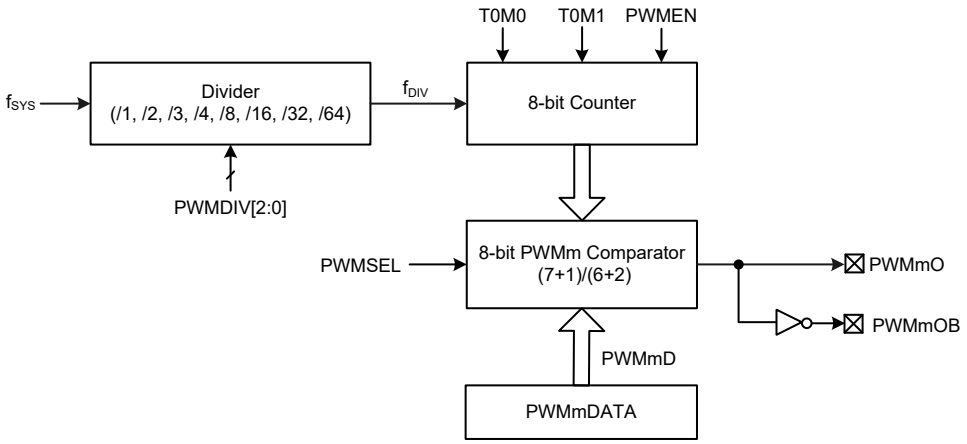
Pulse Width Measurement Mode Timing Chart (T0EG=0)

PWM Mode

To select this mode, bits T0M1 and T0M0 in the PWMC register should be set to "00" respectively. In this mode, the Timer/Event Counter can be used to make PWM waveform with the clock source coming from the internal selected clock source.

The PWMC register and PWMmDATA register are assigned to the Pulse Width Modulator channel. The PWM channel has a data register, PWMmDATA, the content of which is an 8-bit data, abbreviated as PWMmD, representing the overall duty cycle of one modulation cycle of the output waveform. To increase the PWM modulation frequency, each modulation cycle is subdivided into two or four individual modulation subsections, known as the (7+1) bits mode or (6+2) bits mode respectively. The PWM counter clock frequency, f_{DIV}, comes from f_{SYS} or its division. The required mode, clock source selection and the enable/disable control for the PWM channel are selected using the PWMC register. Note that when using the PWM, it is only necessary to write the required value into the PWMmDATA register and select the required mode, clock source and enable/disable control using the PWMC register, the subdivision of the waveform into its sub-modulation cycles is implemented automatically within the microcontroller hardware. Note that the PWMAOB pin is the inverse output of the PWMAO, which can generate a complementary output and supplying more power to connected interfaces such as buzzers.

This method of dividing the original modulation cycle into a further 2 or 4 sub-cycles enables the generation of higher PWM frequencies which allow a wider range of applications to be served. The difference between what is known as the PWM cycle frequency and the PWM modulation frequency should be understood. As the PWM clock is f_{DIV}, and as the PWM value is 8-bit wide, the overall PWM cycle frequency is $f_{DIV}/256$. However, when in the (7+1) mode the PWM modulation frequency will be $f_{DIV}/128$, while the PWM modulation frequency for the (6+2) mode will be $f_{DIV}/64$.



Note: m=A, B or C, and only the PWMAO has the inverter output pin PWMAOB.

PWM Mode Block Diagram

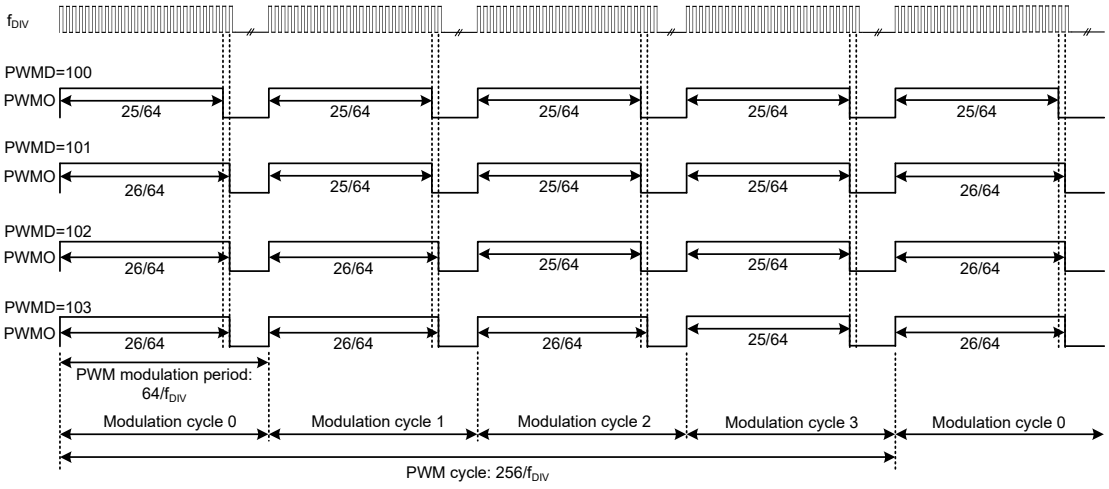
(6+2) Bits PWM Mode Modulation

A (6+2) bits mode PWM cycle is divided into four modulation cycles, which are named as Modulation cycle 0 ~ Modulation cycle 3. Each modulation cycle has 64 PWM input clock periods. In the (6+2) bits PWM mode, the PWMmD is divided into two groups. Group 1 is denoted by DC which is the value of PWMmD bit 7 ~ bit 2. Group 2 is denoted by AC which is the value of PWMmD bit 1 ~ bit 0. The modulation frequency, modulation cycle duty, PWM cycle frequency and PWM cycle duty of the (6+2) bits mode PWM output signals are summarized in the following table.

Modulation Frequency	Modulation Cycle i	Modulation Cycle Duty		PWM Cycle Frequency	PWM Cycle Duty
f_{DIV}/64	i=0~3	i<AC	(DC+1)/64	f_{DIV}/256	PWMmD/256
		i≥AC	DC/64		

(6+2) Bits PWM Mode Summary

The following diagram illustrates the waveforms associated with the (6+2) bits mode of PWM operation. It is important to note how the single PWM cycle is subdivided into 4 individual modulation cycles, numbered from 0~3 and how the AC value is related to the PWM value. The waveforms of PWM outputs are as shown below.



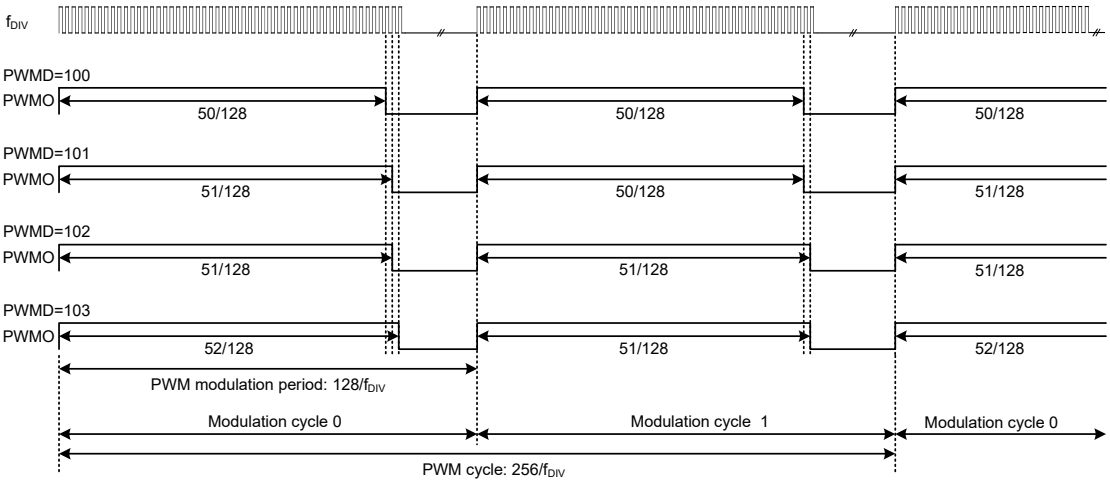
(6+2) Bits PWM Mode Modulation Waveform

(7+1) Bits PWM Mode Modulation

A (7+1) bits mode PWM cycle is divided into two modulation cycles, which is named as Modulation cycle 0 ~ Modulation cycle 1. Each modulation cycle has 128 PWM input clock periods. In the (7+1) bits PWM mode, the PWMmD is divided into two groups. Group 1 is denoted by DC which is the value of PWMmD bit 7 ~ bit 1. Group 2 is denoted by AC which is the value of PWMmD bit 0. The modulation frequency, modulation cycle duty, PWM cycle frequency and PWM cycle duty of the (7+1) bits mode PWM output signals are summarized in the following table.

Modulation Frequency	Modulation Cycle i	Modulation Cycle Duty		PWM Cycle Frequency	PWM Cycle Duty
f_{DIV}/128	i=0~1	i<AC	(DC+1)/128	f_{DIV}/256	PWMmD/256
		i≥AC	DC/128		

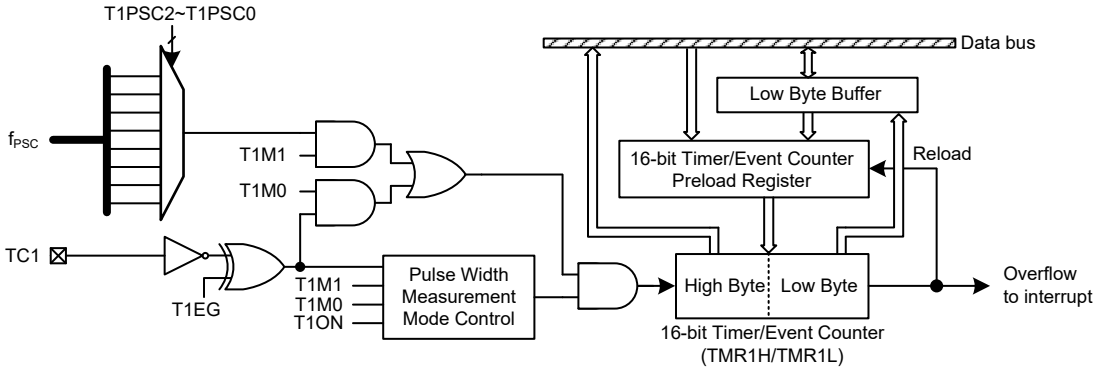
(7+1) Bits PWM Mode Summary



(7+1) Bits PWM Mode Modulation Waveform

16-bit Timer/Event Counter

The provision of the Timer/Event Counter forms an important part of any microcontroller, giving the designer a means of carrying out time related functions. The device contains a 16-bit Timer/Event Counter, which contains a 16-bit programmable count-up counter and the clock may come from an external or internal clock source. As the Timer/Event Counter has three different operating modes, it can be configured to operate as a general timer, an external event counter or a pulse width measurement device.



16-bit Timer/Event Counter Structure

Timer/Event Counter Input Clock Source

The Timer/Event Counter clock source can originate from various sources, an internal clock or an external pin. The internal clock source is used when the timer is in the Timer Mode and Pulse Width Measurement Mode. For the Timer/Event Counter, this internal clock source is sourced from the divided version of Time Base prescaler, which is selected by the T1PSC2~T1PSC0 bits in the TMR1C Timer/Event Control Register.

An external clock source is used when the Timer/Event Counter is in the Event Counter Mode, the clock source is provided on the external TC1 pin. Depending upon the condition of the T1EG bit, each high to low or low to high transition on the external timer pin will increase the counter by one.

Timer/Event Counter Registers

There are two types of registers related to the Timer/Event Counter. The first is the TMR1L/TMR1H register pair that contains the actual value of the timer and into which an initial value can be preloaded. Writing to the TMR1L/TMR1H register will transfer the specified data to the Timer/Event Counter. Reading the TMR1L/TMR1H register will read the contents of the Timer/Event Counter. The second is the TMR1C control register, which is used to define the operating mode, control the counting enable or disable and select the active edge.

Register Name	Bit							
	7	6	5	4	3	2	1	0
TMR1C	T1M1	T1M0	—	T1ON	T1EG	T1PSC2	T1PSC1	T1PSC0
TMR1L	D7	D6	D5	D4	D3	D2	D1	D0
TMR1H	D15	D14	D13	D12	D11	D10	D9	D8

Timer/Event Counter Register List

Timer Register – TMR1L, TMR1H

As the timer/event counter contains an internal 16-bit timer, it requires two data registers to store the value. There are a high byte register, known as TMR1H, and a low byte register, known as TMR1L. The value in the timer register pair increases by one each time an internal clock pulse is received or an external transition occurs on the external timer pin. The timer will count from the initial value loaded by the preload register to the full count of FFFFH, at which point the timer overflows and an internal interrupt signal is generated. The timer value will then be reloaded with the initial preload register value and continue counting.

Note that to achieve a maximum full counting range of FFFFH, the preload register must first be cleared. If the timer/event counter is in an OFF condition and data is written to its preload register, this data will be immediately written into the actual counter. However, if the counter is enabled and counting, any new data written into the preload data register during this period will be remained in the preload register and will not be written into the actual counter until an overflow occurs.

Writing to the TMR1L register will only write the data into an internal lower byte buffer while writing to the TMR1H register will transfer the high byte data and the contents of the lower byte buffer into the TMR1H and TMR1L registers respectively. Therefore the timer/event counter preload register is changed by each write operation to the TMR1H register. Reading from the TMR1H register will latch the contents of the TMR1H and TMR1L counters to the destination and the lower byte buffer respectively, while reading from TMR1L will read the contents of the lower byte buffer.

- **TMR1L Register**

Bit	7	6	5	4	3	2	1	0
Name	D7	D6	D5	D4	D3	D2	D1	D0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	0	0	0	0	0

Bit 7~0 **D7~D0**: Timer preload register low byte

- **TMR1H Register**

Bit	7	6	5	4	3	2	1	0
Name	D15	D14	D13	D12	D11	D10	D9	D8
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	0	0	0	0	0

Bit 7~0 **D15~D8**: Timer preload register high byte

Timer Control Register – TMR1C

The Timer Control Register is known as TMR1C. It is the Timer Control Register together with its corresponding timer register that controls the full operation of the Timer/Event Counter. Before the timer can be used, it is essential that the Timer Control Register is fully programmed with the right data to ensure its correct operation, a process that is normally carried out during program initialisation.

To select which of the three modes the timer is to operate in, namely the Timer Mode, the Event Counter Mode or the Pulse Width Measurement Mode, the T1M1~T1M0 bits in the Timer Control Register must be set to the required logic levels. The timer-on bit T1ON provides the basic on/off control of the respective timer. Setting the bit to high allows the counter to run. Clearing the bit stops the counter. When the internal clock f_{PSC} is used, it can be selected by properly setting the T1PSC2~T1PSC0 bits. The internal clock selection will have no effect if an external clock source is used. If the timer is in the Event Counter or Pulse Width Measurement Mode, the active transition edge type is selected by the logic level of the T1EG bit in the Timer Control Register.

- **TMR1C Register**

Bit	7	6	5	4	3	2	1	0
Name	T1M1	T1M0	—	T1ON	T1EG	T1PSC2	T1PSC1	T1PSC0
R/W	R/W	R/W	—	R/W	R/W	R/W	R/W	R/W
POR	0	0	—	0	1	0	0	0

Bit 7~6 **T1M1~T1M0**: Timer/Event Counter operating mode selection
00: Unused
01: Event Counter Mode
10: Timer Mode
11: Pulse Width Measurement Mode

Bit 5 Unimplemented, read as "0"

Bit 4 **T1ON**: Timer/Event Counter counting enable
0: Disable
1: Enable

Bit 3 **T1EG**: Timer/Event Counter active edge selection
Event Counter Mode
0: Count on rising edge
1: Count on falling edge
Pulse Width Measurement Mode
0: Start counting on falling edge, stop on rising edge
1: Start counting on rising edge, stop on falling edge

Bit 2~0 **T1PSC2~T1PSC0**: Timer internal clock selection
000: $f_{PSC}/2^0$
001: $f_{PSC}/2^1$
010: $f_{PSC}/2^2$
011: $f_{PSC}/2^3$
100: $f_{PSC}/2^4$
101: $f_{PSC}/2^5$
110: $f_{PSC}/2^6$
111: $f_{PSC}/2^7$

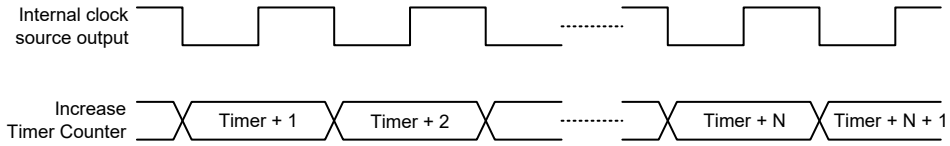
Timer/Event Counter Operating Modes

The Timer/Event Counter can operate in one of three operating modes, Timer Mode, Event Counter Mode or Pulse Width Measurement Mode. The operating mode is selected using the T1M1 and T1M0 bits in the TMR1C register.

Timer Mode

To select this mode, bits T1M1 and T1M0 in the TMR1C register should be set to "10" respectively. In this mode, the Timer/Event Counter can be utilised to measure fixed time intervals, providing an internal interrupt signal each time the Timer/Event Counter overflows.

When operating in this mode the internal clock f_{PSC} is used as the timer clock. The clock source is from the Time Base prescaler, and is selected by the T1PSC2~T1PSC0 bits in the TMR1C register. The timer-on bit T1ON must be set high to enable the timer to run. Each time an internal clock high to low transition occurs, the timer increases by one. When the timer reaches its maximum 16-bit, FFFF Hex, value and overflows, an interrupt request is generated and thetimer will reload the value already loaded into the preload register and continue counting. It should be noted that in the Timer mode, even if the device is in the IDLE/SLEEP mode, if the selected internal clock is still activated the timer will continue to count.



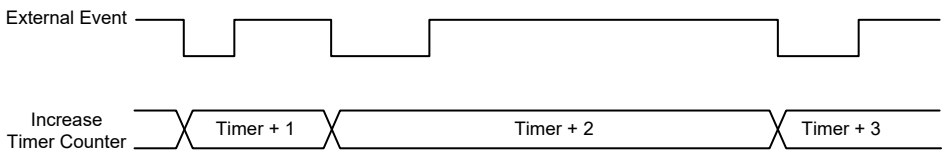
Timer Mode Timing Chart

Event Counter Mode

To select this mode, bits T1M1 and T1M0 in the TMR1C register should be set to "01" respectively. In this mode, a number of externally changing logic events, occurring on the external timer TC1 pin, can be recorded by the Timer/Event Counter.

When operating in this mode, the external timer pin, TC1, is used as the Timer/Event Counter clock source. After the other bits in the Timer Control Register have been properly configured, the enable bit T1ON, can be set high to enable the Timer/Event Counter. If the Active Edge Selection bit, T1EG, is low, the Timer/Event Counter will increase each time the TC1 pin receives a low to high transition. If the T1EG bit is high, the counter will increase each time the TC1 pin receives a high to low transition. When it is full and overflows, an interrupt signal is generated and the Timer/Event Counter will reload the value already loaded into the preload register and continue counting.

As the external timer pin TC1 is pin-shared with other pin functions, the TC1 pin function must first be selected using relevant pin-shared function selection bits. The pin must also be set as an input by setting the corresponding bit in the port control register. It should be noted that in the Event Counter mode, even if the device is in the IDLE/SLEEP Mode, the Timer/Event Counter will continue to record externally changing logic events on the TC1 pin. As a result when the timer overflows it will generate a timer interrupt and corresponding wake-up source.



Event Counter Mode Timing Chart (T1EG=1)

Pulse Width Measurement Mode

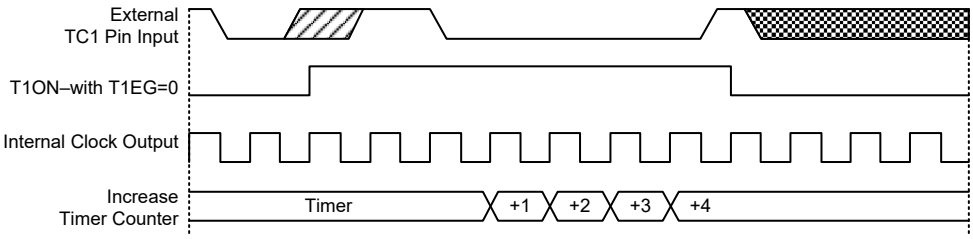
To select this mode, bits T1M1 and T1M0 in the TMR1C register should be set to "11" respectively. In this mode, the Timer/Event Counter can be utilised to measure the width of external pulses applied to the external timer pin.

When operating in this mode the internal clock f_{PSC} is used as the timer clock. The clock source is from the Time Base prescaler, the division ratio is determined by the T1PSC2~T1PSC0 bits in the TMR1C register. After the other bits in the Timer Control Register have been properly configured, the enable bit T1ON, can be set high to enable the Timer/Event Counter, however it will not actually start counting until an active edge is received on the TC1 pin.

If the active Edge Selection bit T1EG is low, once a high to low transition has been received on the TC1 pin, the Timer/Event Counter will start counting based on the internal selected clock source until the TC1 pin returns to its original high level. At this point the enable bit will be automatically reset to zero and the Timer/Event Counter will stop counting. If the Active Edge Selection bit is high, the Timer/Event Counter will begin counting once a low to high transition has been received on the TC1 pin and stop counting when the external timer pin returns to its original low level. As before, the enable bit will then be automatically reset to zero. It is important to note that in the pulse width measurement Mode, the enable bit is automatically reset to zero when the external control signal on the TC1 pin returns to its original level, whereas in the other two modes the enable bit can only be reset to zero under application program control.

The residual value in the Timer/Event Counter, which can now be read by the program, therefore represents the length of the pulse received on the TC1 pin. As the enable bit has now been reset, any further transitions on the external timer pin will be ignored. The timer cannot begin further pulse width measurement until the enable bit is set high again by the application program. In this way, single shot pulse measurements can be easily made. It should be noted that in this mode the Timer/Event Counter is controlled by logical transitions on the external timer pin and not by the logic level. When the Timer/Event Counter is full and overflows, an interrupt signal is generated and the Timer/Event Counter will reload the value already loaded into the preload register and continue counting.

As the external timer pin TC1 is pin-shared with other pin functions, the TC1 pin function must first be selected using relevant pin-shared function selection bits. The pin must also be set as an input by setting the corresponding bit in the port control register. It should be noted that in the Pulse Width Capture mode, even if the device is in the IDLE/SLEEP Mode, the Timer/Event Counter will continue to record externally changing logic events on the TC1 pin if the internal clock source is still activated and the external signal continues to change state. As a result when the timer overflows it will generate a timer interrupt and corresponding wake-up source.



Internal Clock Output is sampled at every falling edge of T1.

Pulse Width Measurement Mode Timing Chart (T1EG=0)

Programming Considerations

When running in the Timer Mode, if the internal system clock is used as the timer clock source, the timer can be synchronised with the overall operation of the microcontroller. In this mode when the timer register is full, the microcontroller will generate an internal interrupt signal directing the program flow to the respective internal interrupt vector. For the Pulse Width Measurement Mode, the internal timer clock is also used as the timer clock source but the timer will only run when the correct logic condition appears on the external timer input pin. As this is an external event and not synchronised with the internal timer clock, the microcontroller will only see this external event when the next timer clock pulse arrives. As a result, there may be small errors in measured values requiring programmers to take this into account during programming. The same applies if the timer is configured to operate in the Event Counter Mode, which again is an external event and not synchronised with the internal timer clock.

When the Timer/Event Counter is read, or if data is written to the preload register, the clock is inhibited to avoid errors, however as this may result in a counting error, it should be taken into account by the programmer. Care must be taken to ensure that the timers are properly initialised before using them for the first time. The associated timer enable bit in the interrupt control register must be properly set otherwise the internal interrupt associated with the timer will remain inactive. The active edge selection, timer operating mode selection and clock source control bits in timer control register must also be correctly issued to ensure the timer is properly configured for the required applications. It is also important to ensure that a desired initial value is first loaded into the timer register before the timer is switched on. After the timer has been initialised the timer can be turned on and off by controlling the enable bit in the timer control register.

When the Timer/Event Counter overflows, its corresponding interrupt request flag in the interrupt control register will be set to generate an interrupt signal. If the Timer/Event Counter interrupt is enabled this will in turn allow program branch to its interrupt vector. However irrespective of whether the interrupt is enabled or not, a Timer/Event Counter overflow will also generate a wake-up signal if the device is in the IDLE/SLEEP mode. This situation may occur if the Timer/Event Counter internal clock source is still activated or if the external signal continues to change state. In such cases, the Timer/Event Counter will continue to count and if an overflow occurs the device will be woken up. To prevent such a wake-up from occurring, the timer interrupt request flag should first be set high before issuing the “HALT” instruction to enter the IDLE/SLEEP mode.

Analog to Digital Converter

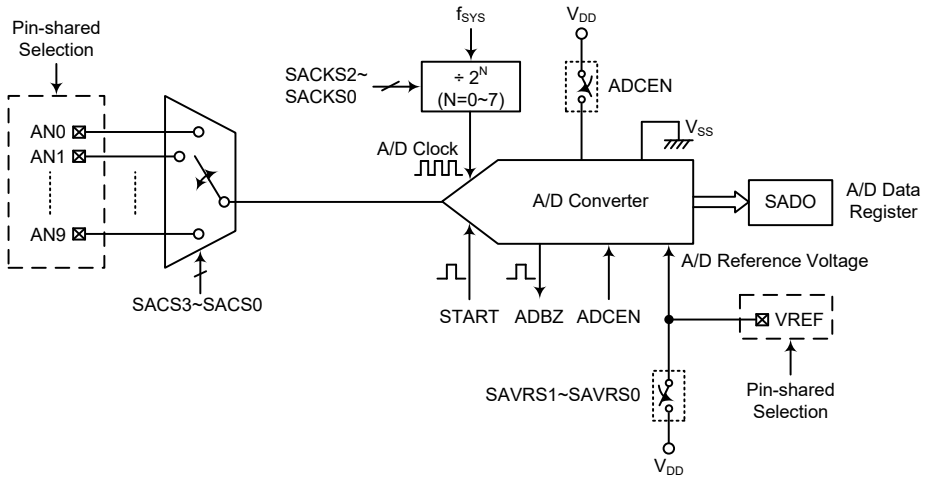
The need to interface to real world analog signals is a common requirement for many electronic systems. However, to properly process these signals by a microcontroller, they must first be converted into digital signals by A/D converters. By integrating the A/D conversion electronic circuitry into the microcontroller, the need for external components is reduced significantly with the corresponding follow-on benefits of lower costs and reduced component space requirements.

A/D Converter Overview

The device contains a multi-channel analog to digital converter which can directly interface to external analog signals, such as that from sensors or other control signals and convert these signals directly into an 8-bit digital value. The external analog signal to be converted is determined by the SACS3~SACS0 bits. When the external analog signal is to be converted, the corresponding pin-shared control bits should first be properly configured and then desired external channel input should be selected using the SACS3~SACS0 bits. More detailed information about the A/D input signal selection will be described in the "A/D converter Control Registers" and "A/D Converter Input Signals" section respectively.

External Input Channels	A/D Signal Select Bits
10: AN0~AN9	SACS3~SACS0

The accompanying block diagram shows the internal structure of the A/D converter together with its associated registers and control bits.



A/D Converter Structure

A/D Converter Register Description

Overall operation of the A/D converter is controlled using five registers. A read only register exists to store the A/D Converter data 8-bit value. The remaining two registers, SADC0 and SADC1, are control registers which set the operating conditions and control function of the A/D converter.

Register Name	Bit							
	7	6	5	4	3	2	1	0
SADO	D7	D6	D5	D4	D3	D2	D1	D0
SADC0	START	ADBZ	ADCEN	—	SACS3	SACS2	SACS1	SACS0
SADC1	—	—	—	SAVRS1	SAVRS0	SACKS2	SACKS1	SACKS0

A/D Converter Register List

A/D Converter Data Register – SADO

As the device contains an internal 8-bit A/D converter, it only requires a data register to store the converted value. This is the register SADO. After the conversion process takes place, the register can be directly read by the microcontroller to obtain the digitised conversion value. D0~D7 are the A/D conversion result data bits. The A/D data registers contents will be cleared if the A/D converter is disabled.

Bit	7	6	5	4	3	2	1	0
Name	D7	D6	D5	D4	D3	D2	D1	D0
R/W	R	R	R	R	R	R	R	R
POR	x	x	x	x	x	x	x	x

A/D Converter Data Register

A/D Converter Control Registers – SADC0, SADC1

To control the function and operation of the A/D converter, two control registers known as SADC0 and SADC1 are provided. These 8-bit registers define functions such as the selection of which analog signal is connected to the internal A/D converter, the digitised data format, the A/D clock source as well as controlling the start function and monitoring the A/D converter busy status. As the device contains only one actual analog to digital converter hardware circuit, each of the external and internal analog signals must be routed to the converter. The SACS3~SACS0 bits in the SADC0 register are used to determine which external channel input is selected to be converted.

The relevant pin-shared function selection bits determine which pins on I/O Ports are used as analog inputs for the A/D converter input and which pins are not to be used as the A/D converter input. When the pin is selected to be an A/D input, its original function whether it is an I/O or other pin-shared function will be removed. In addition, any internal pull-high resistor connected to the pin will be automatically removed if the pin is selected to be an A/D converter input.

- **SADC0 Register**

Bit	7	6	5	4	3	2	1	0
Name	START	ADBZ	ADCEN	—	SACS3	SACS2	SACS1	SACS0
R/W	R/W	R	R/W	—	R/W	R/W	R/W	R/W
POR	0	0	0	—	0	0	0	0

Bit 7 **START**: Start the A/D conversion
0→1→0: Start
This bit is used to initiate an A/D conversion process. The bit is normally low but if set high and then cleared low again, the A/D converter will initiate a conversion process.

Bit 6 **ADBZ**: A/D converter busy flag
0: No A/D conversion is in progress
1: A/D conversion is in progress
This read only flag is used to indicate whether the A/D conversion is in progress or not. When the START bit is set from low to high and then to low again, the ADBZ flag will be set to 1 to indicate that the A/D conversion is initiated. The ADBZ flag will be cleared to 0 after the A/D conversion is complete.

Bit 5 **ADCEN**: A/D converter function enable control
0: Disable
1: Enable
This bit controls the A/D internal function. This bit should be set to one to enable the A/D converter. If the bit is set low, then the A/D converter will be switched off reducing the device power consumption. When the A/D converter function is disabled, the contents of the A/D data register known as SADO will be unchanged.

Bit 4 Unimplemented, read as "0"

Bit 3~0 **SACS3~SACS0**: A/D converter external analog input channel selection
0000: AN0
0001: AN1
0010: AN2
0011: AN3
0100: AN4
0101: AN5
0110: AN6
0111: AN7
1000: AN8
1001: AN9
1010~1111: Non-existed channel, the input will be floating if selected.

• **SADC1 Register**

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	SAVRS1	SAVRS0	SACKS2	SACKS1	SACKS0
R/W	—	—	—	R/W	R/W	R/W	R/W	R/W
POR	—	—	—	0	0	0	0	0

Bit 7~5 Unimplemented, read as

Bit 4~3 **SAVRS1~SAVRS0**: A/D converter reference voltage selection
00: External VREF pin
01: Internal A/D converter power, V_{DD}
1x: External VREF pin
These bits are used to select the A/D converter reference voltage. Care must be taken if the SAVRS1~SAVRS0 bits are set to "01" to select the internal A/D converter power as the reference voltage source. When the internal A/D converter power is selected as the reference voltage, the VREF pin cannot be configured as the reference voltage input by properly configuring the corresponding pin-shared function control bits. Otherwise, the external input voltage on VREF pin will be connected to the internal A/D converter power.

Bit 2~0 **SACKS2~SACKS0**: A/D conversion clock source selection
000: f_{SYS}
001: $f_{SYS}/2$
010: $f_{SYS}/4$
011: $f_{SYS}/8$
100: $f_{SYS}/16$
101: $f_{SYS}/32$
110: $f_{SYS}/64$
111: $f_{SYS}/128$

A/D Converter Reference Voltage

The actual reference voltage supply to the A/D converter can be supplied from the positive power supply, V_{DD}, or an external reference source supplied on pin VREF, determined by the SAVRS1~SAVRS0 bits in the SADC1 register. When the SAVRS1~SAVRS0 bits are set to "01", the A/D converter reference voltage will come from the internal A/D converter power V_{DD}. Otherwise, if the SAVRS1~SAVRS0 bits are set to any other value except "01", the A/D converter reference voltage will come from the VREF pin. When the VREF pin is selected as the reference voltage supply pin, the relevant pin-shared control bits should first be properly configured to enable the VREF pin function as it is pin-shared with other functions. However, if the internal A/D converter power is selected as the reference voltage, the VREF pin must not be configured as the reference voltage input function to avoid the internal connection between the VREF pin to A/D converter power V_{DD}.

A/D Converter Input Signals

All of the external A/D analog input pins are pin-shared with the I/O pins as well as other functions. The corresponding pin-shared function control bits in the PxSn registers determine whether the external input pins are set as A/D converter analog channel inputs or whether they have other functions. If the corresponding pin is set to be an A/D converter analog channel input, the original pin function will be disabled. In this way, pins can be changed under program control to change their function between A/D inputs and other functions. All pull-high resistors, which are set through register programming, will be automatically disconnected if the pins are set as A/D inputs. Note that it is not necessary to first set the A/D pin as an input in the port control register to enable the A/D input as when the relevant A/D input function selection bits enable an A/D input, the status of the port control register will be overridden.

SACS[3:0]	Input Signals	Description
0000~1001	AN0~AN9	External channel analog input ANn
1010~1111	—	Non-existed channel, input is floating

A/D Converter Input Signal Selection

A/D Converter Operation

The START bit in the SADC0 register is used to start the AD conversion. When the microcontroller sets this bit from low to high and then low again, an analog to digital conversion cycle will be initiated.

The ADBZ bit in the SADC0 register is used to indicate whether the analog to digital conversion process is in progress or not. This bit will be automatically set to 1 by the microcontroller after an A/D conversion is successfully initiated. When the A/D conversion is complete, the ADBZ bit will be cleared to 0. In addition, the corresponding A/D interrupt request flag in the interrupt control register will be set, and an internal A/D interrupt signal will be generated. If the A/D interrupt is enabled, this A/D internal interrupt signal will direct the program flow to the associated A/D internal interrupt address for processing. If the A/D internal interrupt is disabled, the microcontroller can poll the ADBZ bit in the SADC0 register to check whether it has been cleared as an alternative method of detecting the end of an A/D conversion cycle.

The clock source for the A/D converter, which originates from the system clock f_{SYS}, can be chosen to be either f_{SYS} or a subdivided version of f_{SYS}. The division ratio value is determined by the SACKS2~SACKS0 bits in the SADC1 register. Although the A/D clock source is determined by the system clock f_{SYS} and by bits SACKS2~SACKS0, there are some limitations on the A/D clock source speed that can be selected. As the recommended range of permissible A/D clock period, t_{ADCK}, is from 0.5μs to 10μs, care must be taken for system clock frequencies. For example, if the system clock operates at a frequency of 8MHz, the SACKS2~SACKS0 bits should not be set to “000”, “001” or “111”. Doing so will give A/D clock periods that are less than the minimum A/D clock period or greater than the maximum A/D clock period which may result in inaccurate A/D conversion values. Refer to the following table for examples, where values marked with an asterisk * show where, special care must be taken, as the values may be beyond the specified A/D Clock Period range.

f_{SYS}	A/D Clock Period (t_{ADCK})							
	SACKS[2:0] = 000 (f_{SYS})	SACKS[2:0] = 001 (f_{SYS}/2)	SACKS[2:0] = 010 (f_{SYS}/4)	SACKS[2:0] = 011 (f_{SYS}/8)	SACKS[2:0] = 100 (f_{SYS}/16)	SACKS[2:0] = 101 (f_{SYS}/32)	SACKS[2:0] = 110 (f_{SYS}/64)	SACKS[2:0] = 111 (f_{SYS}/128)
1MHz	1μs	2μs	4μs	8μs	16μs*	32μs*	64μs*	128μs*
2MHz	500ns	1μs	2μs	4μs	8μs	16μs*	32μs*	64μs*
4MHz	250ns*	500ns	1μs	2μs	4μs	8μs	16μs*	32μs*
8MHz	125ns*	250ns*	500ns	1μs	2μs	4μs	8μs	16μs*

A/D Clock Period Examples

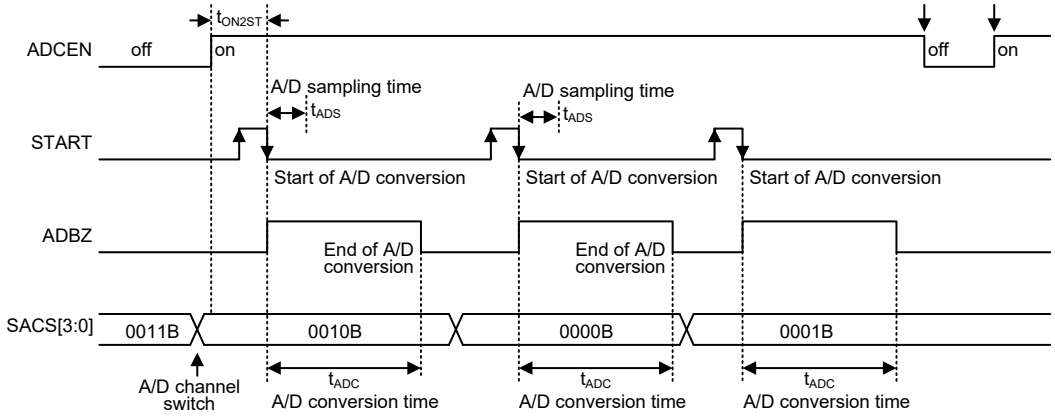
Controlling the power on/off function of the A/D converter circuitry is implemented using the ADCEN bit in the SADC0 register. This bit must be set high to power on the A/D converter. When the ADCEN bit is set high to power on the A/D converter internal circuitry, a certain delay, as indicated in the timing diagram, must be allowed before an A/D conversion is initiated. Even if no pins are selected for use as A/D inputs by configuring the relevant pin-shared control bits, if the ADCEN bit is high, then some power will still be consumed. In power conscious applications it is therefore recommended that the ADCEN is set low to reduce power consumption when the A/D converter function is not being used.

Conversion Rate and Timing Diagram

A complete A/D conversion contains two parts, data sampling and data conversion. The data sampling takes 4 A/D clock periods and the data conversion takes 8 A/D clock periods. Therefore a total of 12 A/D clock periods for an analog signal A/D conversion which is defined as t_{ADC} are necessary.

Maximum single A/D conversion rate = 1 ÷ (A/D clock period × 12)

The accompanying diagram shows graphically the various stages involved in an external channel input signal analog to digital conversion process and its associated timing. After an A/D conversion process has been initiated by the application program, the microcontroller internal hardware will begin to carry out the conversion, during which time the program can continue with other functions. The time taken for an A/D conversion is 12 t_{ADCK} where t_{ADCK} is equal to the A/D clock period.



A/D Conversion Timing – External Channel Input

Summary of A/D Conversion Steps

The following summarises the individual steps that should be executed in order to implement an A/D conversion process.

- Step 1
  Select the required A/D conversion clock by properly programming the SACKS2~SACKS0 bits in the SADC1 register.
- Step 2
  Enable the A/D converter by setting the ADCEN bit in the SADC0 register to one.
- Step 3
  The corresponding pins should be configured as A/D input function by configuring the relevant pin-shared function control bits. The desired analog channel then should be selected by configuring the SACS3~SACS0 bits.
- Step 4
  Select the reference voltage source by configuring the SAVRS1~SAVRS0 bits in the SADC1 register. If the reference voltage is selected from V_{DD}, ensure the VREF pin function not be selected by configuring the PAS0 and PCS0 registers.
- Step 5
  If A/D conversion interrupt is used, the interrupt control registers must be correctly configured to ensure the A/D interrupt function is active. The master interrupt control bit, EMI, and the A/D conversion interrupt control bit, ADE, must both be set high in advance.
- Step 6
  The A/D conversion procedure can now be initialized by setting the START bit from low to high and then low again.
- Step 7
  If A/D conversion is in progress, the ADBZ flag will be set high. After the A/D conversion process is complete, the ADBZ flag will go low and then the output data can be read from SADO register.

Note: When checking for the end of the conversion process, if the method of polling the ADBZ bit in the SADC0 register is used, the interrupt enable step above can be omitted.

Programming Considerations

During microcontroller operations where the A/D converter is not being used, the A/D internal circuitry can be switched off to reduce power consumption, by clearing bit ADCEN to zero in the SADC0 register. When this happens, the internal A/D converter circuits will not consume power irrespective of what analog voltage is applied to their input lines. If the A/D converter input lines are used as normal I/Os, then care must be taken as if the input voltage is not at a valid logic level, then this may lead to some increase in power consumption.

A/D Conversion Function

As the device contains a 8-bit A/D converter, its full-scale converted digitised value is equal to FFH. Since the full-scale analog input value is equal to the actual A/D converter reference voltage, V_{REF}, this gives a single bit analog input value of reference voltage value divided by 256.

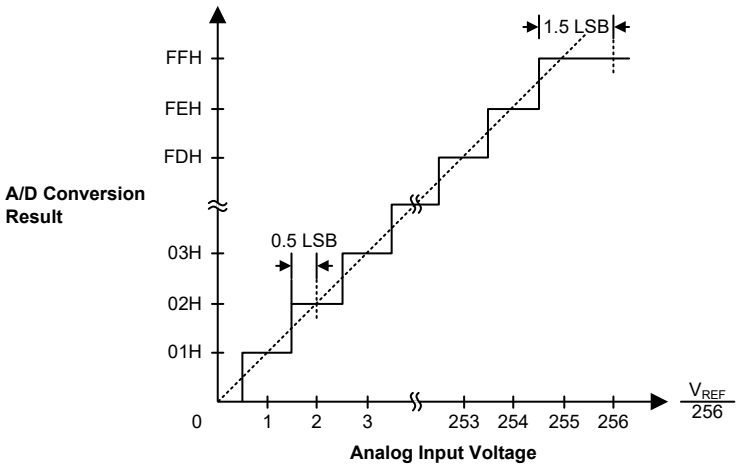
$$1\ LSB = V_{REF} \div 256$$

The A/D Converter input voltage value can be calculated using the following equation:

$$\text{A/D input voltage} = \text{A/D output digital value} \times (V_{REF} \div 256)$$

The diagram shows the ideal transfer function between the analog input value and the digitised output value for the A/D converter. Except for the digitised zero value, the subsequent digitised values will change at a point 0.5 LSB below where they would change without the offset, and the last full scale digitised value will change at a point 1.5 LSB below the V_{REF} level.

Note that here the V_{REF} voltage is the actual A/D converter reference voltage source determined by the SAVRS1~SAVRS0 bits.



Ideal A/D Transfer Function

A/D Conversion Programming Examples

The following two programming examples illustrate how to set and implement an A/D conversion. In the first example, the method of polling the ADBZ bit in the SADC0 register is used to detect when the conversion cycle is complete, whereas in the second example, the A/D interrupt is used to determine when the conversion is complete.

Example 1: using an ADBZ polling method to detect the end of conversion

```
clr ADE                ; disable ADC interrupt
mov a,03h              ; select A/D input signal from external channel, reference voltage
                       ; from VDD, and fSYS/8 as A/D clock
mov SADC1,a
mov a,03h
mov PAS0,a             ; set PAS0 to configure pin AN0
mov a,20h
mov SADC0,a            ; enable the A/D converter and connect AN0 channel to A/D converter
:
start_conversion:
clr START              ; high pulse on start bit to initiate conversion
set START              ; reset A/D
clr START              ; start A/D
polling_EOC:
sz  ADBZ               ; poll the SADC0 register ADBZ bit to detect end of A/D conversion
jmp polling_EOC        ; continue polling
mov a,SADO             ; read byte conversion result value
mov SADO_buffer,a      ; save result to user defined register
:
:
jmp start_conversion   ; start next A/D conversion
```

Example 2: using the interrupt method to detect the end of conversion

```
clr ADE                ; disable ADC interrupt
mov a,03h              ; select A/D input signal from external channel, reference voltage
                       ; from VDD, and fSYS/8 as A/D clock
mov SADC1,a
mov a,03h
mov PAS0,a             ; set PAS0 to configure pin AN0
mov a,20h
mov SADC0,a            ; enable the A/D converter and connect AN0 channel to A/D converter
:
start_conversion:
clr START              ; high pulse on START bit to initiate conversion
set START              ; reset A/D
clr START              ; start A/D
clr ADF                ; clear ADC interrupt request flag
set ADE                ; enable ADC interrupt
set EMI                ; enable global interrupt
:
:
; ADC interrupt service routine
ADC_ISR:
mov acc_stack,a        ; save ACC to user defined memory
mov a,STATUS
mov status_stack,a     ; save STATUS to user defined memory
:
:
mov a,SADO             ; read byte conversion result value
mov SADO_buffer,a      ; save result to user defined register
:
:
EXIT_INT_ISR:
mov a,status_stack
mov STATUS,a           ; restore STATUS from user defined memory
mov a,acc_stack        ; restore ACC from user defined memory
reti
```

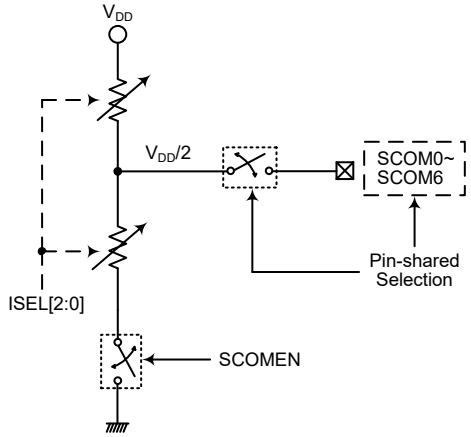
Software Controlled LCD Driver

The device has the capability of driving external LCD panels. The common pins, SCOM0~SCOM6, for LCD driving are pin-shared with certain pins on the I/O ports. The LCD signals (COM) are generated using the application program.

LCD Operation

An external LCD panel can be driven using the device by configuring the I/O pins as common pins. The LCD driver function is controlled using the SCOMC register which in addition to controlling the overall on/off function also controls the R-type bias current on the SCOMn pins. This enables the LCD COM driver to generate the necessary voltage levels, V_{SS}, $V_{DD}/2$ and V_{DD}, for LCD 1/2 bias operation.

The SCOMEN bit in the SCOMC register is the overall master control for the LCD driver. The SCOMn pin is selected to be used for LCD driving by the corresponding pin-shared function selection bits. Note that the corresponding Port Control register does not need to first setup the pins as outputs to enable the LCD driver operation.



Software Controlled LCD Driver Structure

LCD Bias Current Control

The LCD COM driver enables a range of selections to be provided to suit the requirement of the LCD panel which is being used. The bias current choice is implemented using the ISEL2~ISEL0 bits in the SCOMC register. All COM pins are pin-shared with I/O pins and selected as SCOM pins using the corresponding pin-shared function selection bits.

- **SCOMC Register**

Bit	7	6	5	4	3	2	1	0
Name	ISEL2	ISEL1	ISEL0	SCOMEN	—	—	—	—
R/W	R/W	R/W	R/W	R/W	—	—	—	—
POR	0	0	0	0	—	—	—	—

Bit 7~5 **ISEL2~ISEL0**: SCOM typical bias current selection (@V_{DD}=5V)
000: 25μA
001: 50μA
010: 100μA
011: 200μA
100: 5μA
101~111: 12.5μA

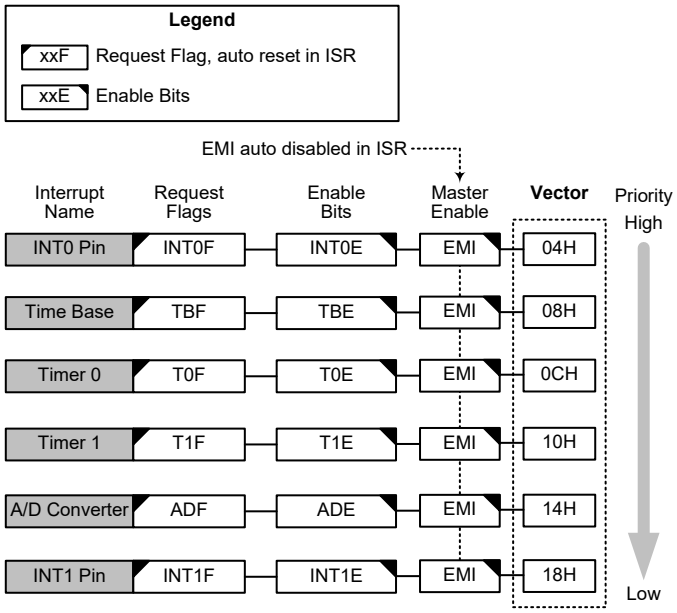
Bit 4 **SCOMEN**: Software controlled LCD Driver enable control
0: Disable
1: Enable
The SCOMn lines can be enabled using the corresponding pin-shared selection bits if the SCOMEN bit is set to 1. When the SCOMEN bit is cleared to 0, then the SCOMn outputs will be fixed at a V_{DD} level. Note that the corresponding pin-shared selection bits should first be properly configured before the SCOMn function is enabled.

Bit 3~0 Unimplemented, read as "0"

Interrupts

Interrupts are an important part of any microcontroller system. When an external event or an internal function such as a Timer requires microcontroller attention, its corresponding interrupt will enforce a temporary suspension of the main program allowing the microcontroller to direct attention to its needs. The device contains several external interrupt and internal interrupt functions. The external interrupt is generated by the action of the external INT0~INT1 pins, while the internal interrupts are generated by internal functions including the Timer/Event Counter and Time Base, etc.

The various interrupt enable bits, together with their associated request flags, are shown in the accompanying diagram with their order of priority. All interrupt sources have their own individual vector.



Interrupt Structure

Interrupt Registers

Overall interrupt control, which basically means the setting of request flags when certain microcontroller conditions occur and the setting of interrupt enable bits by the application program, is controlled by a series of registers, located in the Special Purpose Data Memory, as shown in the accompanying table. The number of registers falls into two categories. The first is the INTC0~INTC1 registers which setup the primary interrupts, the second is the INTEG register to set the external interrupt trigger edge type.

The interrupt register contains a number of enable bits to enable or disable individual registers as well as interrupt flags to indicate the presence of an interrupt request. The naming convention of these follows a specific pattern. First is listed an abbreviated interrupt type, then the (optional) number of that interrupt followed by either an "E" for enable/disable bit or "F" for request flag.

Function	Enable Bit	Request Flag	Notes
Global	EMI	—	—
INTn Pin	INTnE	INTnF	n=0~1
Time Base	TBE	TBF	—
Timer/Event Counter	TnE	TnF	n = 0~1
A/D Converter	ADE	ADF	—

Interrupt Register Bit Naming Conventions

Register Name	Bit							
	7	6	5	4	3	2	1	0
INTEG	—	—	—	—	INT1S1	INT1S0	INT0S1	INT0S0
INTC0	—	T0F	TBF	INT0F	T0E	TBE	INT0E	EMI
INTC1	—	INT1F	ADF	T1F	—	INT1E	ADE	T1E

Interrupt Register List

• INTEG Register

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	INT1S1	INT1S0	INT0S1	INT0S0
R/W	—	—	—	—	R/W	R/W	R/W	R/W
POR	—	—	—	—	0	0	0	0

Bit 7~4 Unimplemented, read as "0"

Bit 3~2 **INT1S1~INT1S0**: Interrupt edge control for INT1 pin
00: Disable
01: Rising edge
10: Falling edge
11: Rising and falling edges

Bit 1~0 **INT0S1~INT0S0**: Interrupt edge control for INT0 pin
00: Disable
01: Rising edge
10: Falling edge
11: Rising and falling edges

• INTC0 Register

Bit	7	6	5	4	3	2	1	0
Name	—	T0F	TBF	INT0F	T0E	TBE	INT0E	EMI
R/W	—	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	—	0	0	0	0	0	0	0

Bit 7 Unimplemented, read as "0"

Bit 6 **T0F**: Timer/Event Counter 0 interrupt request flag
0: No request
1: Interrupt request

Bit 5 **TBF**: Time Base interrupt request flag
0: No request
1: Interrupt request

Bit 4 **INT0F**: INT0 interrupt request flag
0: No request
1: Interrupt request

Bit 3 **T0E**: Timer/Event Counter 0 interrupt control
0: Disable
1: Enable

Bit 2 **TBE**: Time Base interrupt control
0: Disable
1: Enable

Bit 1 **INT0E**: INT0 interrupt control
0: Disable
1: Enable

Bit 0 **EMI**: Global interrupt control
0: Disable
1: Enable

- **INTC1 Register**

Bit	7	6	5	4	3	2	1	0
Name	—	INT1F	ADF	T1F	—	INT1E	ADE	T1E
R/W	—	R/W	R/W	R/W	—	R/W	R/W	R/W
POR	—	0	0	0	—	0	0	0

Bit 7 Unimplemented, read as "0"

Bit 6 **INT1F**: INT1 interrupt request flag
0: No request
1: Interrupt request

Bit 5 **ADF**: A/D Converter interrupt request flag
0: No request
1: Interrupt request

Bit 4 **T1F**: Timer/Event Counter 1 interrupt request flag
0: No request
1: Interrupt request

Bit 3 Unimplemented, read as "0"

Bit 2 **INT1E**: INT1 interrupt control
0: Disable
1: Enable

Bit 1 **ADE**: A/D Converter interrupt control
0: Disable
1: Enable

Bit 0 **T1E**: Timer/Event Counter 1 interrupt control
0: Disable
1: Enable

Interrupt Operation

When the conditions for an interrupt event occur, such as a timer overflow, the relevant interrupt request flag will be set. Whether the request flag actually generates a program jump to the relevant interrupt vector is determined by the condition of the interrupt enable bit. If the enable bit is set high then the program will jump to its relevant vector; if the enable bit is zero then although the interrupt request flag is set an actual interrupt will not be generated and the program will not jump to the relevant interrupt vector. The global interrupt enable bit, if cleared to zero, will disable all interrupts.

When an interrupt is generated, the Program Counter, which stores the address of the next instruction to be executed, will be transferred onto the stack. The Program Counter will then be loaded with a new address which will be the value of the corresponding interrupt vector. The microcontroller will then fetch its next instruction from this interrupt vector. The instruction at this vector will usually be a "JMP" which will jump to another section of program which is known as the interrupt service routine. Here is located the code to control the appropriate interrupt. The interrupt service routine must be terminated with a "RETI", which retrieves the original Program Counter address from the stack and allows the microcontroller to continue with normal execution at the point where the interrupt occurred.

Once an interrupt subroutine is serviced, all the other interrupts will be blocked, as the global interrupt enable bit, EMI bit will be cleared automatically. This will prevent any further interrupt nesting from occurring. However, if other interrupt requests occur during this interval, although the interrupt will not be immediately serviced, the request flag will still be recorded.

If an interrupt requires immediate servicing while the program is already in another interrupt service routine, the EMI bit should be set after entering the routine, to allow interrupt nesting. If the stack is full, the interrupt request will not be acknowledged, even if the related interrupt is enabled, until the Stack Pointer is decremented. If immediate service is desired, the stack must be prevented from becoming full. In case of simultaneous requests, the Interrupt Structure diagram shows the priority that is applied. All of the interrupt request flags when set will wake-up the device if it is in SLEEP or IDLE Mode, however to prevent a wake-up from occurring the corresponding flag should be set before the device is in SLEEP or IDLE Mode.

External Interrupt

The external interrupt is controlled by signal transitions on the INTn pin. An external interrupt request will take place when the external interrupt request flag, INTnF, is set, which will occur when a transition, whose type is chosen by the edge select bits, appears on the external interrupt pin. To allow the program to branch to its respective interrupt vector address, the global interrupt enable bit, EMI, and the external interrupt enable bit, INTnE, must first be set. Additionally the correct interrupt edge type must be selected using the INTEG register to enable the external interrupt function and to choose the trigger edge type. As the external interrupt pin is pin-shared with I/O pin, it can only be configured as external interrupt pin if its external interrupt enable bit in the corresponding interrupt register has been set and the external interrupt pin is selected by the corresponding pin-shared function selection bits. The pin must also be set as an input by setting the corresponding bit in the port control register. When the interrupt is enabled, the stack is not full and the correct transition type appears on the external interrupt pin, a subroutine call to the external interrupt vector, will take place. When the interrupt is serviced, the external interrupt request flag, INTnF, will be automatically reset and the EMI bit will be automatically cleared to disable other interrupts. Note that any pull-high resistor selection on the external interrupt pin will remain valid even if the pin is used as an external interrupt input.

The INTEG register is used to select the type of active edge that will trigger the external interrupt. A choice of either rising or falling or both edge types can be chosen to trigger an external interrupt. Note that the INTEG register can also be used to disable the external interrupt function.

Timer/Event Counter Interrupt

An actual Timer/Event Counter interrupt will take place when the Timer/Event Counter request flag, TnF, is set, which occurs when the Timer/Event Counter overflows. To allow the program to branch to its respective interrupt vector address, the global interrupt enable bit, EMI, and the Timer/Event Counter Interrupt enable bit, TnE, must first be set. When the interrupt is enabled, the stack is not full and the Timer/Event Counter overflows, a subroutine call to its interrupt vector, will take place. When the interrupt is serviced, the Timer/Event Counter Interrupt flag, TnF, will be automatically cleared. The EMI bit will also be automatically cleared to disable other interrupts.

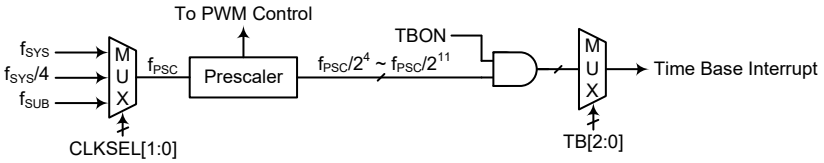
A/D Converter Interrupt

An A/D Converter Interrupt request will take place when the A/D Converter Interrupt request flag, ADF, is set, which occurs when the A/D conversion process finishes. To allow the program to branch to its respective interrupt vector address, the global interrupt enable bit, EMI, and A/D Interrupt enable bit, ADE, must first be set. When the interrupt is enabled, the stack is not full and the A/D conversion process has ended, a subroutine call to the A/D Interrupt vector, will take place. When the A/D Converter Interrupt is serviced, the A/D Interrupt flag, ADF, will be automatically cleared. The EMI bit will also be automatically cleared to disable other interrupts.

Time Base Interrupt

The function of the Time Base Interrupt is to provide regular time signal in the form of an internal interrupt. It is controlled by the overflow signals from the timer function. When this happens its interrupt request flag TBF will be set. To allow the program to branch to its interrupt vector address, the global interrupt enable bit, EMI and Time Base enable bit, TBE, must first be set. When the interrupt is enabled, the stack is not full and the Time Base overflows, a subroutine call to its interrupt vector location will take place. When the interrupt is serviced, the interrupt request flag, TBF, will be automatically reset and the EMI bit will be cleared to disable other interrupts.

The purpose of the Time Base Interrupt is to provide an interrupt signal at fixed time periods. Its clock source, f_{PSC}, originates from the internal clock source f_{SYS}, $f_{SYS}/4$ or f_{SUB} and then passes through a divider, the division ratio of which is selected by programming the appropriate bits in the TBC register to obtain longer interrupt periods whose value ranges. The clock source which in turn controls the Time Base interrupt period is selected using the CLKSEL[1:0] bits in the TBC register. It should be noted that as the Time Base clock source is the same as the Timer/Event Counter clock source, care should be taken when programming.



Time Base Interrupt

• **TBC Register**

Bit	7	6	5	4	3	2	1	0
Name	TBON	—	CLKSEL1	CLKSEL0	—	TB2	TB1	TB0
R/W	R/W	—	R/W	R/W	—	R/W	R/W	R/W
POR	0	—	0	0	—	0	0	0

Bit 7 **TBON**: Time Base control
0: Disable
1: Enable

Bit 6 Unimplemented, read as "0"

Bit 5~4 **CLKSEL1~CLKSEL0**: Prescaler clock source f_{PSC} selection
00: f_{SYS}
01: $f_{SYS}/4$
1x: f_{SUB}

Bit 3 Unimplemented, read as "0"

Bit 2~0 **TB2~TB0**: Time Base Time-out period selection
000: $2^4/f_{PSC}$
001: $2^5/f_{PSC}$
010: $2^6/f_{PSC}$
011: $2^7/f_{PSC}$
100: $2^8/f_{PSC}$
101: $2^9/f_{PSC}$
110: $2^{10}/f_{PSC}$
111: $2^{11}/f_{PSC}$

Interrupt Wake-up Function

Each of the interrupt functions has the capability of waking up the microcontroller when in the SLEEP or IDLE Mode. A wake-up is generated when an interrupt request flag changes from low to high and is independent of whether the interrupt is enabled or not. Therefore, even though the device is in the SLEEP or IDLE Mode and its system oscillator stopped, situations such as external edge transitions on the external interrupt pin may cause their respective interrupt flag to be set high and consequently generate an interrupt. Care must therefore be taken if spurious wake-up situations are to be avoided. If an interrupt wake-up function is to be disabled then the corresponding interrupt request flag should be set high before the device enters the SLEEP or IDLE Mode. The interrupt enable bits have no effect on the interrupt wake-up function.

Programming Considerations

By disabling the relevant interrupt enable bits, a requested interrupt can be prevented from being serviced, however, once an interrupt request flag is set, it will remain in this condition in the interrupt register until the corresponding interrupt is serviced or until the request flag is cleared by the application program.

It is recommended that programs do not use the "CALL" instruction within the interrupt service subroutine. Interrupts often occur in an unpredictable manner or need to be serviced immediately. If only one stack is left and the interrupt is not well controlled, the original control sequence will be damaged once a CALL subroutine is executed in the interrupt subroutine.

Every interrupt has the capability of waking up the microcontroller when it is in the SLEEP or IDLE Mode, the wake up being generated when the interrupt request flag changes from low to high. If it is required to prevent a certain interrupt from waking up the microcontroller then its respective request flag should be first set high before enter SLEEP or IDLE Mode.

As only the Program Counter is pushed onto the stack, then when the interrupt is serviced, if the contents of the accumulator, status register or other registers are altered by the interrupt service program, their contents should be saved to the memory at the beginning of the interrupt service routine.

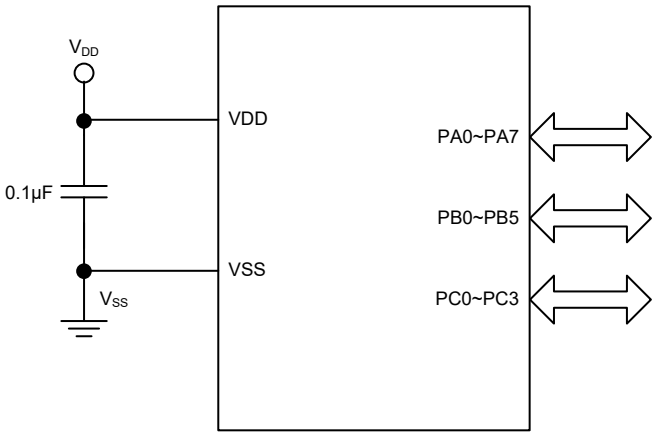
To return from an interrupt subroutine, either a RET or RETI instruction may be executed. The RETI instruction in addition to executing a return to the main program also automatically sets the EMI bit high to allow further interrupts. The RET instruction however only executes a return to the main program leaving the EMI bit in its present zero state and therefore disabling the execution of further interrupts.

Configuration Options

Configuration options refer to certain options within the MCU that are programmed into the device during the programming process. During the development process, these options are selected using the IDE software development tools. As these options are programmed into the device using the hardware programming tools, once they are selected they cannot be changed later using the application program.All options must be defined for proper system function, the details of which are shown in the table.

No.	Options
I/O Pin-Shared Options	
1	RES pin reset function selection: 0: Always I/O or other function 1: RES pin
LVR Options	
2	LVR function selection: 0: Enable 1: Disable
3	LVR voltage selection: 00: 1.9V 01: 2.1V 10: 3.15V 11: 4.2V

Application Circuits



Instruction Set

Introduction

Central to the successful operation of any microcontroller is its instruction set, which is a set of program instruction codes that directs the microcontroller to perform certain operations. In the case of the microcontroller, a comprehensive and flexible set of over 60 instructions is provided to enable programmers to implement their application with the minimum of programming overheads.

For easier understanding of the various instruction codes, they have been subdivided into several functional groupings.

Instruction Timing

Most instructions are implemented within one instruction cycle. The exceptions to this are branch, call, or table read instructions where two instruction cycles are required. One instruction cycle is equal to 4 system clock cycles, therefore in the case of an 8MHz system oscillator, most instructions would be implemented within 0.5μs and branch or call instructions would be implemented within 1μs. Although instructions which require one more cycle to implement are generally limited to the JMP, CALL, RET, RETI and table read instructions, it is important to realize that any other instructions which involve manipulation of the Program Counter Low register or PCL will also take one more cycle to implement. As instructions which change the contents of the PCL will imply a direct jump to that new address, one more cycle will be required. Examples of such instructions would be "CLR PCL" or "MOV PCL, A". For the case of skip instructions, it must be noted that if the result of the comparison involves a skip operation then this will also take one more cycle, if no skip is involved then only one cycle is required.

Moving and Transferring Data

The transfer of data within the microcontroller program is one of the most frequently used operations. Making use of three kinds of MOV instructions, data can be transferred from registers to the Accumulator and vice-versa as well as being able to move specific immediate data directly into the Accumulator. One of the most important data transfer applications is to receive data from the input ports and transfer data to the output ports.

Arithmetic Operations

The ability to perform certain arithmetic operations and data manipulation is a necessary feature of most microcontroller applications. Within the microcontroller instruction set are a range of add and subtract instruction mnemonics to enable the necessary arithmetic to be carried out. Care must be taken to ensure correct handling of carry and borrow data when results exceed 255 for addition and less than 0 for subtraction. The increment and decrement instructions INC, INCA, DEC and DECA provide a simple means of increasing or decreasing by a value of one of the values in the destination specified.

Logical and Rotate Operation

The standard logical operations such as AND, OR, XOR and CPL all have their own instruction within the microcontroller instruction set. As with the case of most instructions involving data manipulation, data must pass through the Accumulator which may involve additional programming steps. In all logical data operations, the zero flag may be set if the result of the operation is zero. Another form of logical data manipulation comes from the rotate instructions such as RR, RL, RRC and RLC which provide a simple means of rotating one bit right or left. Different rotate instructions exist depending on program requirements. Rotate instructions are useful for serial port programming applications where data can be rotated from an internal register into the Carry bit from where it can be examined and the necessary serial bit set high or low. Another application which rotate data operations are used is to implement multiplication and division calculations.

Branches and Control Transfer

Program branching takes the form of either jumps to specified locations using the JMP instruction or to a subroutine using the CALL instruction. They differ in the sense that in the case of a subroutine call, the program must return to the instruction immediately when the subroutine has been carried out. This is done by placing a return instruction "RET" in the subroutine which will cause the program to jump back to the address right after the CALL instruction. In the case of a JMP instruction, the program simply jumps to the desired location. There is no requirement to jump back to the original jumping off point as in the case of the CALL instruction. One special and extremely useful set of branch instructions are the conditional branches. Here a decision is first made regarding the condition of a certain data memory or individual bits. Depending upon the conditions, the program will continue with the next instruction or skip over it and jump to the following instruction. These instructions are the key to decision making and branching within the program perhaps determined by the condition of certain input switches or by the condition of internal data bits.

Bit Operations

The ability to provide single bit operations on Data Memory is an extremely flexible feature of all microcontrollers. This feature is especially useful for output port bit programming where individual bits or port pins can be directly set high or low using either the "SET [m].i" or "CLR [m].i" instructions respectively. The feature removes the need for programmers to first read the 8-bit output port, manipulate the input data to ensure that other bits are not changed and then output the port with the correct new data. This read-modify-write process is taken care of automatically when these bit operation instructions are used.

Table Read Operations

Data storage is normally implemented by using registers. However, when working with large amounts of fixed data, the volume involved often makes it inconvenient to store the fixed data in the Data Memory. To overcome this problem, al microcontrollers allow an area of Program Memory to be set as a table where data can be directly stored. A set of easy to use instructions provides the means by which this fixed data can be referenced and retrieved from the Program Memory.

Other Operations

In addition to the above functional instructions, a range of other instructions also exist such as the "HALT" instruction for Power-down operations and instructions to control the operation of the Watchdog Timer for reliable program operations under extreme electric or electromagnetic environments. For their relevant operations, refer to the functional related sections.

Instruction Set Summary

The following table depicts a summary of the instruction set categorised according to function and can be consulted as a basic instruction reference using the following listed conventions.

Table Conventions

x: Bits immediate data
m: Data Memory address
A: Accumulator
i: 0~7 number of bits
addr: Program memory address

Mnemonic	Description	Cycles	Flag Affected
Arithmetic			
ADD A,[m]	Add Data Memory to ACC	1	Z, C, AC, OV
ADDM A,[m]	Add ACC to Data Memory	1Note	Z, C, AC, OV
ADD A,x	Add immediate data to ACC	1	Z, C, AC, OV
ADC A,[m]	Add Data Memory to ACC with Carry	1	Z, C, AC, OV
ADCM A,[m]	Add ACC to Data memory with Carry	1Note	Z, C, AC, OV
SUB A,x	Subtract immediate data from the ACC	1	Z, C, AC, OV
SUB A,[m]	Subtract Data Memory from ACC	1	Z, C, AC, OV
SUBM A,[m]	Subtract Data Memory from ACC with result in Data Memory	1Note	Z, C, AC, OV
SBC A,[m]	Subtract Data Memory from ACC with Carry	1	Z, C, AC, OV
SBCM A,[m]	Subtract Data Memory from ACC with Carry, result in Data Memory	1Note	Z, C, AC, OV
DAA [m]	Decimal adjust ACC for Addition with result in Data Memory	1Note	C
Logic Operation			
AND A,[m]	Logical AND Data Memory to ACC	1	Z
OR A,[m]	Logical OR Data Memory to ACC	1	Z
XOR A,[m]	Logical XOR Data Memory to ACC	1	Z
ANDM A,[m]	Logical AND ACC to Data Memory	1Note	Z
ORM A,[m]	Logical OR ACC to Data Memory	1Note	Z
XORM A,[m]	Logical XOR ACC to Data Memory	1Note	Z
AND A,x	Logical AND immediate Data to ACC	1	Z
OR A,x	Logical OR immediate Data to ACC	1	Z
XOR A,x	Logical XOR immediate Data to ACC	1	Z
CPL [m]	Complement Data Memory	1Note	Z
CPLA [m]	Complement Data Memory with result in ACC	1	Z
Increment & Decrement			
INCA [m]	Increment Data Memory with result in ACC	1	Z
INC [m]	Increment Data Memory	1Note	Z
DECA [m]	Decrement Data Memory with result in ACC	1	Z
DEC [m]	Decrement Data Memory	1Note	Z
Rotate			
RRA [m]	Rotate Data Memory right with result in ACC	1	None
RR [m]	Rotate Data Memory right	1Note	None
RRCA [m]	Rotate Data Memory right through Carry with result in ACC	1	C
RRC [m]	Rotate Data Memory right through Carry	1Note	C
RLA [m]	Rotate Data Memory left with result in ACC	1	None
RL [m]	Rotate Data Memory left	1Note	None
RLCA [m]	Rotate Data Memory left through Carry with result in ACC	1	C
RLC [m]	Rotate Data Memory left through Carry	1Note	C

Mnemonic	Description	Cycles	Flag Affected
Data Move			
MOV A,[m]	Move Data Memory to ACC	1	None
MOV [m],A	Move ACC to Data Memory	1Note	None
MOV A,x	Move immediate data to ACC	1	None
Bit Operation			
CLR [m].i	Clear bit of Data Memory	1Note	None
SET [m].i	Set bit of Data Memory	1Note	None
Branch Operation			
JMP addr	Jump unconditionally	2	None
SZ [m]	Skip if Data Memory is zero	1Note	None
SZA [m]	Skip if Data Memory is zero with data movement to ACC	1Note	None
SZ [m].i	Skip if bit i of Data Memory is zero	1Note	None
SNZ [m].i	Skip if bit i of Data Memory is not zero	1Note	None
SIZ [m]	Skip if increment Data Memory is zero	1Note	None
SDZ [m]	Skip if decrement Data Memory is zero	1Note	None
SIZA [m]	Skip if increment Data Memory is zero with result in ACC	1Note	None
SDZA [m]	Skip if decrement Data Memory is zero with result in ACC	1Note	None
CALL addr	Subroutine call	2	None
RET	Return from subroutine	2	None
RET A,x	Return from subroutine and load immediate data to ACC	2	None
RETI	Return from interrupt	2	None
Table Read Operation			
TABRD [m]	Read table (specific page or current page) to TBLH and Data Memory	2Note	None
TABRDL [m]	Read table (last page) to TBLH and Data Memory	2Note	None
Miscellaneous			
NOP	No operation	1	None
CLR [m]	Clear Data Memory	1Note	None
SET [m]	Set Data Memory	1Note	None
CLR WDT	Clear Watchdog Timer	1	TO, PDF
SWAP [m]	Swap nibbles of Data Memory	1Note	None
SWAPA [m]	Swap nibbles of Data Memory with result in ACC	1	None
HALT	Enter power down mode	1	TO, PDF

Note: 1. For skip instructions, if the result of the comparison involves a skip then two cycles are required, if no skip takes place only one cycle is required.

2. Any instruction which changes the contents of the PCL will also require 2 cycles for execution.

Instruction Definition

ADC A,[m] Add Data Memory to ACC with Carry

Description: The contents of the specified Data Memory, Accumulator and the carry flag are added. The result is stored in the Accumulator.

Operation: ACC ← ACC + [m] + C

Affected flag(s): OV, Z, AC, C

ADCM A,[m] Add ACC to Data Memory with Carry

Description: The contents of the specified Data Memory, Accumulator and the carry flag are added. The result is stored in the specified Data Memory.

Operation: [m] ← ACC + [m] + C

Affected flag(s): OV, Z, AC, C

ADD A,[m] Add Data Memory to ACC

Description: The contents of the specified Data Memory and the Accumulator are added. The result is stored in the Accumulator.

Operation: ACC ← ACC + [m]

Affected flag(s): OV, Z, AC, C

ADD A,x Add immediate data to ACC

Description: The contents of the Accumulator and the specified immediate data are added. The result is stored in the Accumulator.

Operation: ACC ← ACC + x

Affected flag(s): OV, Z, AC, C

ADDM A,[m] Add ACC to Data Memory

Description: The contents of the specified Data Memory and the Accumulator are added. The result is stored in the specified Data Memory.

Operation: [m] ← ACC + [m]

Affected flag(s): OV, Z, AC, C

AND A,[m] Logical AND Data Memory to ACC

Description: Data in the Accumulator and the specified Data Memory perform a bitwise logical AND operation. The result is stored in the Accumulator.

Operation: ACC ← ACC "AND" [m]

Affected flag(s): Z

AND A,x Logical AND immediate data to ACC

Description: Data in the Accumulator and the specified immediate data perform a bit wise logical AND operation. The result is stored in the Accumulator.

Operation: ACC ← ACC "AND" x

Affected flag(s): Z

ANDM A,[m] Logical AND ACC to Data Memory

Description: Data in the specified Data Memory and the Accumulator perform a bitwise logical AND operation. The result is stored in the Data Memory.

Operation: [m] ← ACC "AND" [m]

Affected flag(s): Z

CALL addr Subroutine call

Description: Unconditionally calls a subroutine at the specified address. The Program Counter then increments by 1 to obtain the address of the next instruction which is then pushed onto the stack. The specified address is then loaded and the program continues execution from this new address. As this instruction requires an additional operation, it is a two cycle instruction.

Operation:
Stack ← Program Counter + 1
Program Counter ← addr

Affected flag(s): None

CLR [m] Clear Data Memory

Description: Each bit of the specified Data Memory is cleared to 0.

Operation: [m] ← 00H

Affected flag(s): None

CLR [m].i Clear bit of Data Memory

Description: Bit i of the specified Data Memory is cleared to 0.

Operation: [m].i ← 0

Affected flag(s): None

CLR WDT Clear Watchdog Timer

Description: The TO, PDF flags and the WDT are all cleared.

Operation:
WDT cleared
TO ← 0
PDF ← 0

Affected flag(s): TO, PDF

CPL [m] Complement Data Memory

Description: Each bit of the specified Data Memory is logically complemented (1′s complement). Bits which previously contained a 1 are changed to 0 and vice versa.

Operation: [m] ← $\overline{[m]}$

Affected flag(s): Z

CPLA [m] Complement Data Memory with result in ACC

Description: Each bit of the specified Data Memory is logically complemented (1′s complement). Bits which previously contained a 1 are changed to 0 and vice versa. The complemented result is stored in the Accumulator and the contents of the Data Memory remain unchanged.

Operation: ACC ← $\overline{[m]}$

Affected flag(s): Z

DAA [m] Decimal-Adjust ACC for addition with result in Data Memory

Description: Convert the contents of the Accumulator value to a BCD (Binary Coded Decimal) value resulting from the previous addition of two BCD variables. If the low nibble is greater than 9 or if AC flag is set, then a value of 6 will be added to the low nibble. Otherwise the low nibble remains unchanged. If the high nibble is greater than 9 or if the C flag is set, then a value of 6 will be added to the high nibble. Essentially, the decimal conversion is performed by adding 00H, 06H, 60H or 66H depending on the Accumulator and flag conditions. Only the C flag may be affected by this instruction which indicates that if the original BCD sum is greater than 100, it allows multiple precision decimal addition.

Operation:
[m] ← ACC + 00H or
[m] ← ACC + 06H or
[m] ← ACC + 60H or
[m] ← ACC + 66H

Affected flag(s): C

DEC [m] Decrement Data Memory

Description: Data in the specified Data Memory is decremented by 1.

Operation: [m] ← [m] − 1

Affected flag(s): Z

DECA [m] Decrement Data Memory with result in ACC

Description: Data in the specified Data Memory is decremented by 1. The result is stored in the Accumulator. The contents of the Data Memory remain unchanged.

Operation: ACC ← [m] − 1

Affected flag(s): Z

HALT Enter power down mode

Description: This instruction stops the program execution and turns off the system clock. The contents of the Data Memory and registers are retained. The WDT and prescaler are cleared. The power down flag PDF is set and the WDT time-out flag TO is cleared.

Operation: TO ← 0
PDF ← 1

Affected flag(s): TO, PDF

INC [m] Increment Data Memory

Description: Data in the specified Data Memory is incremented by 1.

Operation: [m] ← [m] + 1

Affected flag(s): Z

INCA [m] Increment Data Memory with result in ACC

Description: Data in the specified Data Memory is incremented by 1. The result is stored in the Accumulator. The contents of the Data Memory remain unchanged.

Operation: ACC ← [m] + 1

Affected flag(s): Z

JMP addr Jump unconditionally

Description: The contents of the Program Counter are replaced with the specified address. Program execution then continues from this new address. As this requires the insertion of a dummy instruction while the new address is loaded, it is a two cycle instruction.

Operation: Program Counter ← addr

Affected flag(s): None

MOV A,[m] Move Data Memory to ACC

Description: The contents of the specified Data Memory are copied to the Accumulator.

Operation: ACC ← [m]

Affected flag(s): None

MOV A,x Move immediate data to ACC

Description: The immediate data specified is loaded into the Accumulator.

Operation: ACC ← x

Affected flag(s): None

MOV [m],A Move ACC to Data Memory

Description: The contents of the Accumulator are copied to the specified Data Memory.

Operation: [m] ← ACC

Affected flag(s): None

NOP	No operation
Description	No operation is performed. Execution continues with the next instruction.
Operation	No operation
Affected flag(s)	None

OR A,[m]	Logical OR Data Memory to ACC
Description	Data in the Accumulator and the specified Data Memory perform a bitwise logical OR operation. The result is stored in the Accumulator.
Operation	ACC ← ACC ″OR″ [m]
Affected flag(s)	Z

OR A,x	Logical OR immediate data to ACC
Description	Data in the Accumulator and the specified immediate data perform a bitwise logical OR operation. The result is stored in the Accumulator.
Operation	ACC ← ACC ″OR″ x
Affected flag(s)	Z

ORM A,[m]	Logical OR ACC to Data Memory
Description	Data in the specified Data Memory and the Accumulator perform a bitwise logical OR operation. The result is stored in the Data Memory.
Operation	[m] ← ACC ″OR″ [m]
Affected flag(s)	Z

RET	Return from subroutine
Description	The Program Counter is restored from the stack. Program execution continues at the restored address.
Operation	Program Counter ← Stack
Affected flag(s)	None

RET A,x	Return from subroutine and load immediate data to ACC
Description	The Program Counter is restored from the stack and the Accumulator loaded with the specified immediate data. Program execution continues at the restored address.
Operation	Program Counter ← Stack ACC ← x
Affected flag(s)	None

RETI	Return from interrupt
Description	The Program Counter is restored from the stack and the interrupts are re-enabled by setting the EMI bit. EMI is the master interrupt global enable bit. If an interrupt was pending when the RETI instruction is executed, the pending Interrupt routine will be processed before returning to the main program.
Operation	Program Counter ← Stack EMI ← 1
Affected flag(s)	None

RL [m]	Rotate Data Memory left
Description	The contents of the specified Data Memory are rotated left by 1 bit with bit 7 rotated into bit 0.
Operation	[m].(i+1) ← [m].i; (i=0~6) [m].0 ← [m].7
Affected flag(s)	None

RLA [m] Rotate Data Memory left with result in ACC

Description: The contents of the specified Data Memory are rotated left by 1 bit with bit 7 rotated into bit 0. The rotated result is stored in the Accumulator and the contents of the Data Memory remain unchanged.

Operation:
ACC.(i+1) ← [m].i; (i=0~6)
ACC.0 ← [m].7

Affected flag(s): None

RLC [m] Rotate Data Memory left through Carry

Description: The contents of the specified Data Memory and the carry flag are rotated left by 1 bit. Bit 7 replaces the Carry bit and the original carry flag is rotated into bit 0.

Operation:
[m].(i+1) ← [m].i; (i=0~6)
[m].0 ← C
C ← [m].7

Affected flag(s): C

RLCA [m] Rotate Data Memory left through Carry with result in ACC

Description: Data in the specified Data Memory and the carry flag are rotated left by 1 bit. Bit 7 replaces the Carry bit and the original carry flag is rotated into the bit 0. The rotated result is stored in the Accumulator and the contents of the Data Memory remain unchanged.

Operation:
ACC.(i+1) ← [m].i; (i=0~6)
ACC.0 ← C
C ← [m].7

Affected flag(s): C

RR [m] Rotate Data Memory right

Description: The contents of the specified Data Memory are rotated right by 1 bit with bit 0 rotated into bit 7.

Operation:
[m].i ← [m].(i+1); (i=0~6)
[m].7 ← [m].0

Affected flag(s): None

RRA [m] Rotate Data Memory right with result in ACC

Description: Data in the specified Data Memory is rotated right by 1 bit with bit 0 rotated into bit 7. The rotated result is stored in the Accumulator and the contents of the Data Memory remain unchanged.

Operation:
ACC.i ← [m].(i+1); (i=0~6)
ACC.7 ← [m].0

Affected flag(s): None

RRC [m] Rotate Data Memory right through Carry

Description: The contents of the specified Data Memory and the carry flag are rotated right by 1 bit. Bit 0 replaces the Carry bit and the original carry flag is rotated into bit 7.

Operation:
[m].i ← [m].(i+1); (i=0~6)
[m].7 ← C
C ← [m].0

Affected flag(s): C

RRCA [m] Rotate Data Memory right through Carry with result in ACC

Description: Data in the specified Data Memory and the carry flag are rotated right by 1 bit. Bit 0 replaces the Carry bit and the original carry flag is rotated into bit 7. The rotated result is stored in the Accumulator and the contents of the Data Memory remain unchanged.

Operation:
ACC.i ← [m].(i+1); (i=0~6)
ACC.7 ← C
C ← [m].0

Affected flag(s): C

SBC A,[m] Subtract Data Memory from ACC with Carry

Description: The contents of the specified Data Memory and the complement of the carry flag are subtracted from the Accumulator. The result is stored in the Accumulator. Note that if the result of subtraction is negative, the C flag will be cleared to 0, otherwise if the result is positive or zero, the C flag will be set to 1.

Operation: ACC ← ACC − [m] − $\overline{C}$

Affected flag(s): OV, Z, AC, C

SBCM A,[m] Subtract Data Memory from ACC with Carry and result in Data Memory

Description: The contents of the specified Data Memory and the complement of the carry flag are subtracted from the Accumulator. The result is stored in the Data Memory. Note that if the result of subtraction is negative, the C flag will be cleared to 0, otherwise if the result is positive or zero, the C flag will be set to 1.

Operation: [m] ← ACC − [m] − $\overline{C}$

Affected flag(s): OV, Z, AC, C

SDZ [m] Skip if decrement Data Memory is 0

Description: The contents of the specified Data Memory are first decremented by 1. If the result is 0 the following instruction is skipped. As this requires the insertion of a dummy instruction while the next instruction is fetched, it is a two cycle instruction. If the result is not 0 the program proceeds with the following instruction.

Operation:
[m] ← [m] − 1
Skip if [m]=0

Affected flag(s): None

SDZA [m] Skip if decrement Data Memory is zero with result in ACC

Description: The contents of the specified Data Memory are first decremented by 1. If the result is 0, the following instruction is skipped. The result is stored in the Accumulator but the specified Data Memory contents remain unchanged. As this requires the insertion of a dummy instruction while the next instruction is fetched, it is a two cycle instruction. If the result is not 0, the program proceeds with the following instruction.

Operation:
ACC ← [m] − 1
Skip if ACC=0

Affected flag(s): None

SET [m] Set Data Memory

Description: Each bit of the specified Data Memory is set to 1.

Operation: [m] ← FFH

Affected flag(s): None

SET [m].i Set bit of Data Memory

Description: Bit i of the specified Data Memory is set to 1.

Operation: [m].i ← 1

Affected flag(s): None

SIZ [m] Skip if increment Data Memory is 0

Description: The contents of the specified Data Memory are first incremented by 1. If the result is 0, the following instruction is skipped. As this requires the insertion of a dummy instruction while the next instruction is fetched, it is a two cycle instruction. If the result is not 0 the program proceeds with the following instruction.

Operation: [m] ← [m] + 1
Skip if [m]=0

Affected flag(s): None

SIZA [m] Skip if increment Data Memory is zero with result in ACC

Description: The contents of the specified Data Memory are first incremented by 1. If the result is 0, the following instruction is skipped. The result is stored in the Accumulator but the specified Data Memory contents remain unchanged. As this requires the insertion of a dummy instruction while the next instruction is fetched, it is a two cycle instruction. If the result is not 0 the program proceeds with the following instruction.

Operation: ACC ← [m] + 1
Skip if ACC=0

Affected flag(s): None

SNZ [m].i Skip if bit i of Data Memory is not 0

Description: If bit i of the specified Data Memory is not 0, the following instruction is skipped. As this requires the insertion of a dummy instruction while the next instruction is fetched, it is a two cycle instruction. If the result is 0 the program proceeds with the following instruction.

Operation: Skip if [m].i ≠ 0

Affected flag(s): None

SUB A,[m] Subtract Data Memory from ACC

Description: The specified Data Memory is subtracted from the contents of the Accumulator. The result is stored in the Accumulator. Note that if the result of subtraction is negative, the C flag will be cleared to 0, otherwise if the result is positive or zero, the C flag will be set to 1.

Operation: ACC ← ACC − [m]

Affected flag(s): OV, Z, AC, C

SUBM A,[m] Subtract Data Memory from ACC with result in Data Memory

Description: The specified Data Memory is subtracted from the contents of the Accumulator. The result is stored in the Data Memory. Note that if the result of subtraction is negative, the C flag will be cleared to 0, otherwise if the result is positive or zero, the C flag will be set to 1.

Operation: [m] ← ACC − [m]

Affected flag(s): OV, Z, AC, C

SUB A,x Subtract immediate data from ACC

Description: The immediate data specified by the code is subtracted from the contents of the Accumulator. The result is stored in the Accumulator. Note that if the result of subtraction is negative, the C flag will be cleared to 0, otherwise if the result is positive or zero, the C flag will be set to 1.

Operation: ACC ← ACC − x

Affected flag(s): OV, Z, AC, C

SWAP [m] Swap nibbles of Data Memory

Description: The low-order and high-order nibbles of the specified Data Memory are interchanged.

Operation: [m].3~[m].0 ↔ [m].7~[m].4

Affected flag(s): None

SWAPA [m] Swap nibbles of Data Memory with result in ACC

Description: The low-order and high-order nibbles of the specified Data Memory are interchanged. The result is stored in the Accumulator. The contents of the Data Memory remain unchanged.

Operation:
ACC.3~ACC.0 ← [m].7~[m].4
ACC.7~ACC.4 ← [m].3~[m].0

Affected flag(s): None

SZ [m] Skip if Data Memory is 0

Description: The contents of the specified Data Memory are read out and then written to the specified Data Memory again. If the contents of the specified Data Memory is 0, the following instruction is skipped. As this requires the insertion of a dummy instruction while the next instruction is fetched, it is a two cycle instruction. If the result is not 0 the program proceeds with the following instruction.

Operation: Skip if [m]=0

Affected flag(s): None

SZA [m] Skip if Data Memory is 0 with data movement to ACC

Description: The contents of the specified Data Memory are copied to the Accumulator. If the value is zero, the following instruction is skipped. As this requires the insertion of a dummy instruction while the next instruction is fetched, it is a two cycle instruction. If the result is not 0 the program proceeds with the following instruction.

Operation:
ACC ← [m]
Skip if [m]=0

Affected flag(s): None

SZ [m].i Skip if bit i of Data Memory is 0

Description: If bit i of the specified Data Memory is 0, the following instruction is skipped. As this requires the insertion of a dummy instruction while the next instruction is fetched, it is a two cycle instruction. If the result is not 0, the program proceeds with the following instruction.

Operation: Skip if [m].i=0

Affected flag(s): None

TABRD [m] Read table (specific page or current page) to TBLH and Data Memory

Description: The low byte of the program code addressed by the table pointer (TBHP and TBLP or only TBLP if no TBHP) is moved to the specified Data Memory and the high byte moved to TBLH.

Operation:
[m] ← program code (low byte)
TBLH ← program code (high byte)

Affected flag(s): None

TABRDL [m] Read table (last page) to TBLH and Data Memory

Description: The low byte of the program code (last page) addressed by the table pointer (TBLP) is moved to the specified Data Memory and the high byte moved to TBLH.

Operation:
[m] ← program code (low byte)
TBLH ← program code (high byte)

Affected flag(s): None

XOR A,[m] Logical XOR Data Memory to ACC

Description: Data in the Accumulator and the specified Data Memory perform a bitwise logical XOR operation. The result is stored in the Accumulator.

Operation: ACC ← ACC ″XOR″ [m]

Affected flag(s): Z

XORM A,[m]	Logical XOR ACC to Data Memory
Description	Data in the specified Data Memory and the Accumulator perform a bitwise logical XOR operation. The result is stored in the Data Memory.
Operation	[m] ← ACC ″XOR″ [m]
Affected flag(s)	Z

XOR A,x	Logical XOR immediate data to ACC
Description	Data in the Accumulator and the specified immediate data perform a bitwise logical XOR operation. The result is stored in the Accumulator.
Operation	ACC ← ACC ″XOR″ x
Affected flag(s)	Z

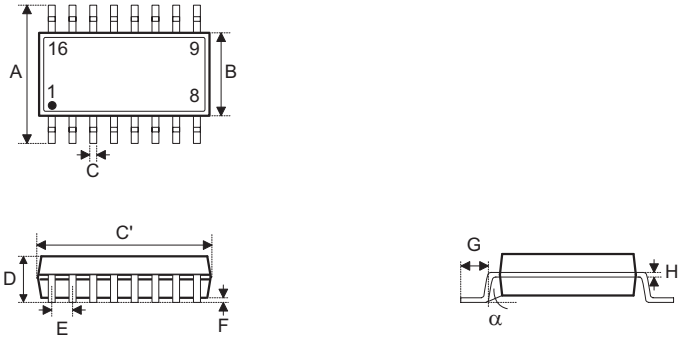
Package Information

Note that the package information provided here is for consultation purposes only. As this information may be updated at regular intervals users are reminded to consult the Website for the latest version of the Package/Carton Information.

Additional supplementary information with regard to packaging is listed below. Click on the relevant section to be transferred to the relevant website page.

- Package Information (include Outline Dimensions, Product Tape and Reel Specifications)
- The Operation Instruction of Packing Materials
- Carton information

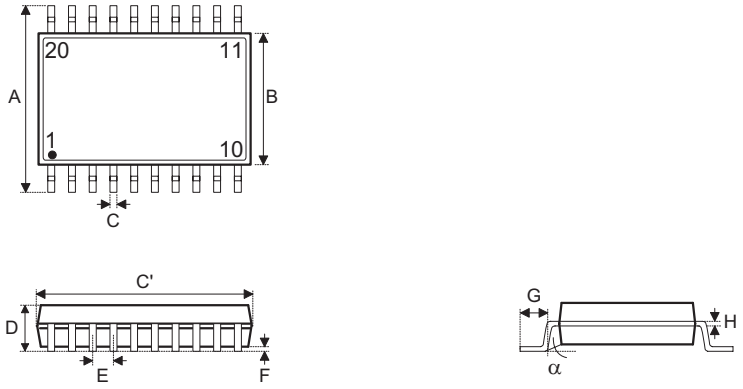
16-pin NSOP (150mil) Outline Dimensions



Symbol	Dimensions in inch		
	Min.	Nom.	Max.
A		0.236 BSC	
B		0.154 BSC	
C	0.012	—	0.020
C’		0.390 BSC	
D	—	—	0.069
E		0.050 BSC	
F	0.004	—	0.010
G	0.016	—	0.050
H	0.004	—	0.010
α	0°	—	8°

Symbol	Dimensions in mm		
	Min.	Nom.	Max.
A		6.00 BSC	
B		3.90 BSC	
C	0.31	—	0.51
C’		9.90 BSC	
D	—	—	1.75
E		1.27 BSC	
F	0.10	—	0.25
G	0.40	—	1.27
H	0.10	—	0.25
α	0°	—	8°

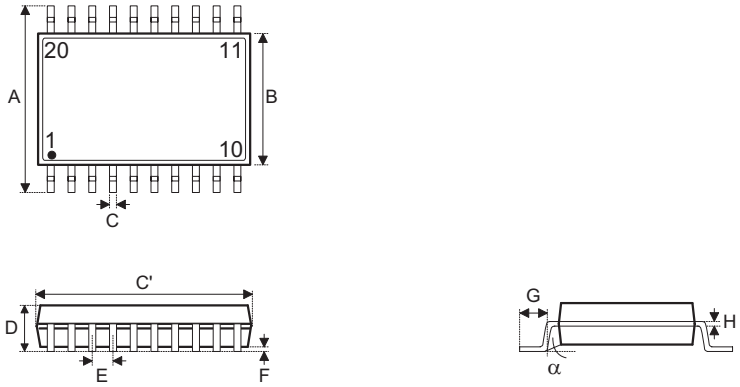
20-pin NSOP (150mil) Outline Dimensions



Symbol	Dimensions in inch		
	Min.	Nom.	Max.
A	0.228	0.236	0.244
B	0.146	0.154	0.161
C	0.009	—	0.012
C’	0.382	0.390	0.398
D	—	—	0.069
E	0.032 BSC		
F	0.002	—	0.009
G	0.020	—	0.031
H	0.008	—	0.010
α	0°	—	8°

Symbol	Dimensions in mm		
	Min.	Nom.	Max.
A	5.80	6.00	6.20
B	3.70	3.90	4.10
C	0.23	—	0.30
C’	9.70	9.90	10.10
D	—	—	1.75
E	0.80 BSC		
F	0.05	—	0.23
G	0.50	—	0.80
H	0.21	—	0.25
α	0°	—	8°

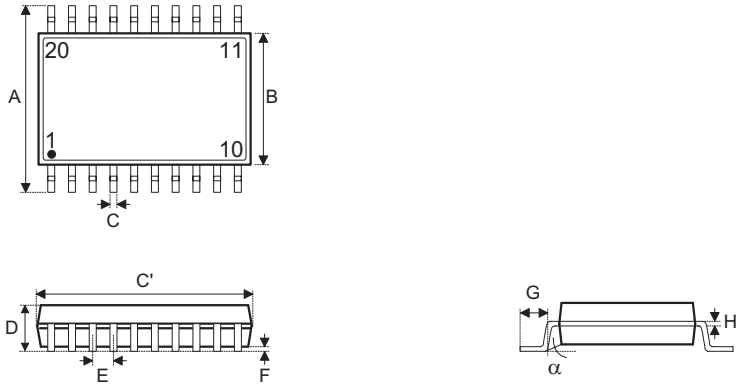
20-pin SOP (300mil) Outline Dimensions



Symbol	Dimensions in inch		
	Min.	Nom.	Max.
A		0.406 BSC	
B		0.295 BSC	
C	0.012	—	0.020
C’		0.504 BSC	
D	—	—	0.104
E		0.050 BSC	
F	0.004	—	0.012
G	0.016	—	0.050
H	0.008	—	0.013
α	0°	—	8°

Symbol	Dimensions in mm		
	Min.	Nom.	Max.
A		10.30 BSC	
B		7.50 BSC	
C	0.31	—	0.51
C’		12.80 BSC	
D	—	—	2.65
E		1.27 BSC	
F	0.10	—	0.30
G	0.40	—	1.27
H	0.20	—	0.33
α	0°	—	8°

20-pin SSOP (150mil) Outline Dimensions



Symbol	Dimensions in inch		
	Min.	Nom.	Max.
A		0.236 BSC	
B		0.154 BSC	
C	0.008	—	0.012
C’		0.341 BSC	
D	—	—	0.069
E		0.025 BSC	
F	0.004	—	0.0098
G	0.016	—	0.05
H	0.004	—	0.01
α	0°	—	8°

Symbol	Dimensions in mm		
	Min.	Nom.	Max.
A		6.000 BSC	
B		3.900 BSC	
C	0.20	—	0.30
C’		8.660 BSC	
D	—	—	1.75
E		0.635 BSC	
F	0.10	—	0.25
G	0.41	—	1.27
H	0.10	—	0.25
α	0°	—	8°