HOLTEK

A/D Flash MCU with EEPROM

HT66F0184

Revision: V1.20 Date: December 30, 2019

Table of Contents

Features **6**
CPU Features 6
Peripheral Features 6
General Description **7**
Block Diagram **7**
Pin Assignment **8**
Pin Description **8**
Absolute Maximum Ratings **11**
D.C. Characteristics **12**
Operating Voltage Characteristics 12
Standby Current Characteristics 12
Operating Current Characteristics 12
A.C. Characteristics **13**
High Speed Internal Oscillator – HIRC – Frequency Accuracy 13
Operating Frequency Characteristic Curves 13
System Start Up Time Characteristics 13
Input/Output Characteristics **14**
Memory Characteristics **14**
LVR Electrical Characteristics **15**
A/D Converter Electrical Characteristics **15**
LCD Driver Electrical Characteristics **15**
Power-on Reset Characteristics **16**
System Architecture **16**
Clocking and Pipelining 16
Program Counter 17
Stack 18
Arithmetic and Logic Unit – ALU 18
Flash Program Memory **19**
Structure 19
Special Vectors 19
Look-up Table 19
Table Program Example 20
In Circuit Programming – ICP 20
On-Chip Debug Support – OCDS 21
Data Memory **22**
Structure 22
General Purpose Data Memory 22
Special Purpose Data Memory 22

Special Function Register Description 24
Indirect Addressing Registers – IAR0, IAR1 24
Memory Pointers – MP0, MP1 24
Bank Pointer – BP 25
Accumulator – ACC 25
Program Counter Low Register – PCL 25
Look-up Table Registers – TBLP, TBHP, TBLH 25
Status Register – STATUS 26

Emulated EEPROM Data Memory 27
Emulated EEPROM Data Memory Structure 27
Emulated EEPROM Registers 28
Erasing the Emulated EEPROM 30
Writing Data to the Emulated EEPROM 31
Reading Data from the Emulated EEPROM 31
Programming Considerations 31

Oscillator Configuration 33

Operating Modes and System Clocks 33
System Clocks 33
System Operation Modes 34
Control Register 35
Operating Mode Switching 35
Standby Current Considerations 37
Wake-up 37

Watchdog Timer 38
Watchdog Timer Clock Source 38
Watchdog Timer Control Register 38
Watchdog Timer Operation 39

Reset and Initialisation 40
Reset Functions 40
Reset Initial Conditions 42

Input/Output Ports 45
Pull-high Resistors 45
Port A Wake-up 46
I/O Port Control Registers 46
Pin-shared Functions 46
I/O Pin Structures 48
Programming Considerations 49

Timer Modules – TM 49
Introduction 49
TM Operation 50
TM Clock Source 50
TM Interrupts 50
TM External Pins 50
Programming Considerations 51

Compact Type TM – CTM **52**
Compact Type TM Operation 52
Compact Type TM Register Description 52
Compact Type TM Operating Modes 56
Standard Type TM – STM **62**
Standard TM Operation 62
Standard Type TM Register Description 62
Standard Type TM Operation Modes 67
Analog to Digital Converter **76**
A/D Converter Overview 76
A/D Converter Register Description 76
A/D Converter Reference Voltage 78
A/D Converter Input Signals 78
A/D Converter Operation 79
Conversion Rate and Timing Diagram 79
Summary of A/D Conversion Steps 80
Programming Considerations 81
A/D Conversion Function 81
A/D Conversion Programming Examples 81
Software Controlled LCD Driver **83**
LCD Operation 83
LCD Control Registers 84
Interrupts **88**
Interrupt Registers 88
Interrupt Operation 91
External Interrupt 92
Multi-function Interrupts 93
A/D Converter Interrupt 93
Timer Module Interrupts 93
Interrupt Wake-up Function 93
Programming Considerations 94
Application Circuits **94**
Instruction Set **95**
Introduction 95
Instruction Timing 95
Moving and Transferring Data 95
Arithmetic Operations 95
Logical and Rotate Operation 96
Branches and Control Transfer 96
Bit Operations 96
Table Read Operations 96
Other Operations 96

Instruction Set Summary **97**
Table Conventions 97
Instruction Definition **99**
Package Information **108**
24-pin SOP (300mil) Outline Dimensions 109
28-pin SOP (300mil) Outline Dimensions 110
24-pin SSOP (150mil) Outline Dimensions 111
28-pin SSOP (150mil) Outline Dimensions 112

Features

CPU Features

- Operating Voltage
  - f_{SYS}=8MHz: 2.2V~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 Type
  - Internal High Speed 8MHz RC – HIRC
- Multi-mode operation: FAST, IDLE and SLEEP
- Fully integrated internal oscillator requires no external components
- All instructions executed in one or two instruction cycles
- Table read instructions
- 63 powerful instructions
- 6-level subroutine nesting
- Bit manipulation instruction

Peripheral Features

- Flash Program Memory: 4K×15
- RAM Data Memory: 192×8
- Emulated EEPROM Memory: 32×15
- Watchdog Timer function
- 26 bidirectional I/O lines
- Software controlled LCD driver function, support 18 SEG×6 COM (Max.) LCD display
  - Output lines: 12 SCOM/SSEG + 12 SSEG
  - Bias level: 1/3 bias
  - Bias type: R type
- One external interrupt line shared with I/O pins
- Multiple Timer Modules for time measure, input capture, compare match output, PWM output function or single pulse output function
- 4 external channels 10-bit resolution A/D converter
- Low voltage reset function
- Package type: 24/28-pin SOP/SSOP

General Description

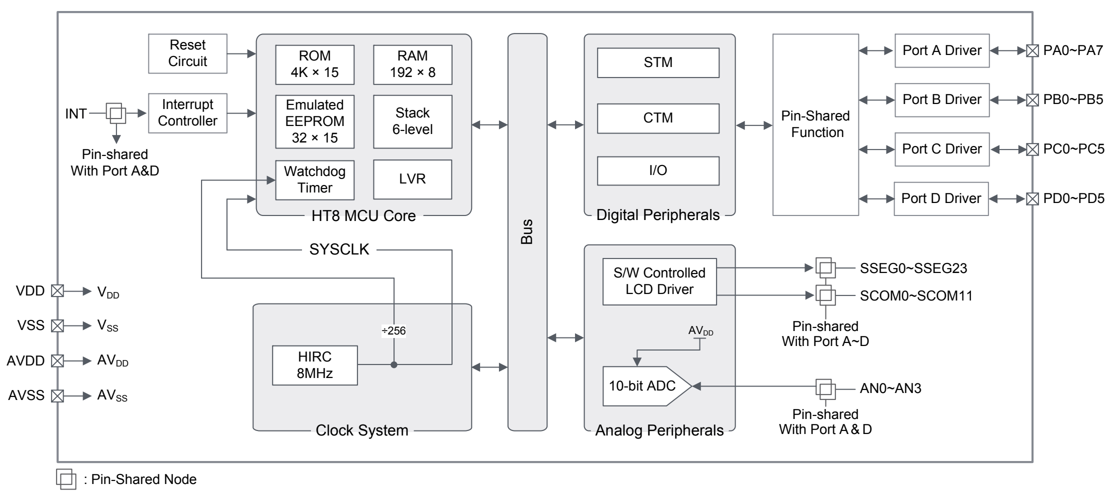
The device is a Flash Memory type 8-bit high performance RISC architecture microcontroller. Offering users the convenience of Flash Memory multi-programming features, the device also includes a wide range of functions and features. Other memory includes an area of RAM Data Memory as well as an area of Emulated EEPROM memory for storage of non-volatile data such as serial numbers, calibration data etc.

Analog features include a multi-channel 10-bit A/D converter and a software controlled LCD driver. Multiple and extremely flexible Timer Modules provide timing, pulse generation and PWM generation functions. An internal Watchdog Timer and Low Voltage Reset function coupled with excellent noise immunity and ESD protection ensures that reliable operation is maintained in hostile electrical environments.

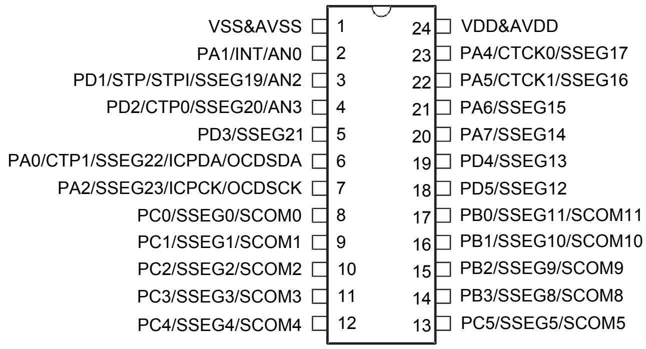
A fully integrated high speed oscillator is provided as the system oscillator which requires no external components for its implementation. The ability to operate and switch dynamically between a range of operating modes using different clock gives users the ability to optimize microcontroller operation and minimize power consumption.

The inclusion of flexible I/O programming features, external interrupt 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.

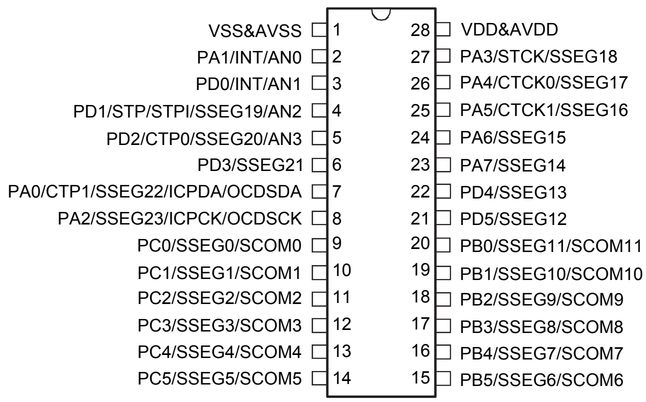
Block Diagram



Pin Assignment



HT66F0184/HT66V0184
24 SOP-A/SSOP-A



HT66F0184/HT66V0184
28 SOP-A/SSOP-A

Note: 1. The desired pin-shared function is determined by the corresponding pin-shared control bits.
2. VDD&AVDD means the VDD and AVDD are double bonding; VSS&AVSS means the VSS and AVSS are double bonding.
3. The OCDSDA and OCDSCK pins are the OCDS dedicated pins and only available for the EV chip with the part number HT66V0184.
4. For the less pin count package type there will be unbounded pins which should be properly configured to avoid unwanted power consumption resulting from floating input conditions. Refer to the "Standby Current Considerations" and "Input/Output Ports" sections.

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/CTP1/SSEG22/ICPDA/OCDSDA	PA0	PAWU PAPU PASR	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up
	CTP1	PASR	—	CMOS	CTM1 output
	SSEG22	SLCDC5	—	AN	Software controlled LCD Segment output
	ICPDA	—	ST	CMOS	ICP Data/Address pin
	OCDSDA	—	ST	CMOS	OCDS Data/Address pin, for EV chip only.

Pin Name	Function	OPT	I/T	O/T	Description
PA1/INT/AN0	PA1	PAWU PAPU PASR	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up
	INT	IFS PASR INTEG INTC0	ST	—	External interrupt input
	AN0	PASR	AN	—	A/D Converter channel 0
PA2/SSEG23/ICPCK/ OCDSCK	PA2	PAWU PAPU	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up
	SSEG23	SLCDC5	—	AN	Software controlled LCD Segment output
	ICPCK	—	ST	CMOS	ICP Clock pin
	OCDSCK	—	ST	—	OCDS Clock pin, for EV chip only.
PA3/STCK/SSEG18	PA3	PAWU PAPU	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up
	STCK	—	ST	—	STM input
	SSEG18	SLCDC5	—	AN	Software controlled LCD Segment output
PA4/CTCK0/SSEG17	PA4	PAWU PAPU	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up
	CTCK0	—	ST	—	CTM0 input
	SSEG17	SLCDC5	—	AN	Software controlled LCD Segment output
PA5/CTCK1/SSEG16	PA5	PAWU PAPU	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up
	CTCK1	—	ST	—	CTM1 input
	SSEG16	SLCDC5	—	AN	Software controlled LCD Segment output
PA6/SSEG15	PA6	PAWU PAPU	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up
	SSEG15	SLCDC4	—	AN	Software controlled LCD Segment output
PA7/SSEG14	PA7	PAWU PAPU	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up
	SSEG14	SLCDC4	—	AN	Software controlled LCD Segment output
PB0/SSEG11/SCOM11	PB0	PBPU	ST	CMOS	General purpose I/O. Register enabled pull-up
	SSEG11	SLCDC2 SLCDC4	—	AN	Software controlled LCD Segment output
	SCOM11	SLCDC2 SLCDC4	—	AN	Software controlled LCD COM output
PB1/SSEG10/SCOM10	PB1	PBPU	ST	CMOS	General purpose I/O. Register enabled pull-up
	SSEG10	SLCDC2 SLCDC4	—	AN	Software controlled LCD Segment output
	SCOM10	SLCDC2 SLCDC4	—	AN	Software controlled LCD COM output
PB2/SSEG9/SCOM9	PB2	PBPU	ST	CMOS	General purpose I/O. Register enabled pull-up
	SSEG9	SLCDC2 SLCDC4	—	AN	Software controlled LCD Segment output
	SCOM9	SLCDC2 SLCDC4	—	AN	Software controlled LCD COM output
PB3/SSEG8/SCOM8	PB3	PBPU	ST	CMOS	General purpose I/O. Register enabled pull-up
	SSEG8	SLCDC2 SLCDC4	—	AN	Software controlled LCD Segment output
	SCOM8	SLCDC2 SLCDC4	—	AN	Software controlled LCD COM output

Pin Name	Function	OPT	I/T	O/T	Description
PB4/SSEG7/SCOM7	PB4	PBPU	ST	CMOS	General purpose I/O. Register enabled pull-up
	SSEG7	SLCDC1 SLCDC3	—	AN	Software controlled LCD Segment output
	SCOM7	SLCDC1 SLCDC3	—	AN	Software controlled LCD COM output
PB5/SSEG6/SCOM6	PB5	PBPU	ST	CMOS	General purpose I/O. Register enabled pull-up
	SSEG6	SLCDC1 SLCDC3	—	AN	Software controlled LCD Segment output
	SCOM6	SLCDC1 SLCDC3	—	AN	Software controlled LCD COM output
PC0/SSEG0/SCOM0	PC0	PCPU	ST	CMOS	General purpose I/O. Register enabled pull-up
	SSEG0	SLCDC1 SLCDC3	—	AN	Software controlled LCD Segment output
	SCOM0	SLCDC1 SLCDC3	—	AN	Software controlled LCD COM output
PC1/SSEG1/SCOM1	PC1	PCPU	ST	CMOS	General purpose I/O. Register enabled pull-up
	SSEG1	SLCDC1 SLCDC3	—	AN	Software controlled LCD Segment output
	SCOM1	SLCDC1 SLCDC3	—	AN	Software controlled LCD COM output
PC2/SSEG2/SCOM2	PC2	PCPU	ST	CMOS	General purpose I/O. Register enabled pull-up
	SSEG2	SLCDC1 SLCDC3	—	AN	Software controlled LCD Segment output
	SCOM2	SLCDC1 SLCDC3	—	AN	Software controlled LCD COM output
PC3/SSEG3/SCOM3	PC3	PCPU	ST	CMOS	General purpose I/O. Register enabled pull-up
	SSEG3	SLCDC1 SLCDC3	—	AN	Software controlled LCD Segment output
	SCOM3	SLCDC1 SLCDC3	—	AN	Software controlled LCD COM output
PC4/SSEG4/SCOM4	PC4	PCPU	ST	CMOS	General purpose I/O. Register enabled pull-up
	SSEG4	SLCDC1 SLCDC3	—	AN	Software controlled LCD Segment output
	SCOM4	SLCDC1 SLCDC3	—	AN	Software controlled LCD COM output
PC5/SSEG5/SCOM5	PC5	PCPU	ST	CMOS	General purpose I/O. Register enabled pull-up
	SSEG5	SLCDC1 SLCDC3	—	AN	Software controlled LCD Segment output
	SCOM5	SLCDC1 SLCDC3	—	AN	Software controlled LCD COM output
PD0/INT/AN1	PD0	PDPU PDSR	ST	CMOS	General purpose I/O. Register enabled pull-up
	INT	IFS PDSR INTEG INTC0	ST	—	External interrupt input
	AN1	PDSR	AN	—	A/D Converter channel 1
PD1/STP/STPI/SSEG19/AN2	PD1	PDPU PDSR	ST	CMOS	General purpose I/O. Register enabled pull-up
	STP	PDSR	—	CMOS	STM output
	STPI	PDSR	ST	—	STM capture input
	SSEG19	SLCDC5	—	AN	Software controlled LCD Segment output
	AN2	PDSR	AN	—	A/D Converter channel 2

Pin Name	Function	OPT	I/T	O/T	Description
PD2/CTP0/SSEG20/AN3	PD2	PDPU PDSR	ST	CMOS	General purpose I/O. Register enabled pull-up
	CTP0	PDSR	—	CMOS	CTM0 output
	SSEG20	SLCDC5	—	AN	Software controlled LCD Segment output
	AN3	PDSR	AN	—	A/D Converter channel 3
PD3/SSEG21	PD3	PDPU	ST	CMOS	General purpose I/O. Register enabled pull-up
	SSEG21	SLCDC5	—	AN	Software controlled LCD Segment output
PD4/SSEG13	PD4	PDPU	ST	CMOS	General purpose I/O. Register enabled pull-up
	SSEG13	SLCDC4	—	AN	Software controlled LCD Segment output
PD5/SSEG12	PD5	PDPU	ST	CMOS	General purpose I/O. Register enabled pull-up
	SSEG12	SLCDC4	—	AN	Software controlled LCD Segment output
VDD&AVDD*	VDD	—	PWR	—	Positive power supply
	AVDD	—	PWR	—	Analog positive power supply
VSS&AVSS*	VSS	—	PWR	—	Ground
	AVSS	—	PWR	—	Analog ground

Legend: I/T: Input type;
O/T: Output type;
OPT: Optional by register option;
PWR: Power;
AN: Analog signal;
ST: Schmitt Trigger input;
CMOS: CMOS output;
SSEG: Software controlled LCD SEG;
SCOM: Software controlled LCD COM.
*: The AVDD pin is internal bonded together with the VDD pin while the AVSS pin is internally bonded together with the VSS pin.

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 .... -50°C to 125°C

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

I_{OL} Total 80mA

I_{OH} 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 this 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 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	2.2	—	5.5	V

Standby Current Characteristics

Ta=25°C, unless otherwise specify

Symbol	Standby Mode	Test Conditions		Min.	Typ.	Max.	Max. @85°C	Unit
		V_{DD}	Conditions					
I_{STB}	SLEEP Mode	2.2V	WDT off	—	0.08	0.12	1.40	μA
		3V		—	0.08	0.12	1.40	
		5V		—	0.15	0.29	2.20	
		2.2V	WDT on, f_{SYS}=8MHz	—	130	250	330	μA
		3V		—	210	350	450	
		5V		—	450	800	960	
	IDLE0 Mode	2.2V	f_{SUB} on, f_{SYS}=8MHz	—	188	300	380	μA
		3V		—	260	400	500	
		5V		—	500	800	960	
	IDLE1 Mode	2.2V	f_{SYS} 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.
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}	FAST Mode – HIRC	2.2V	f_{SYS}=8MHz	—	0.6	1.0	mA
		3V		—	0.8	1.2	mA
		5V		—	1.6	2.4	mA

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.
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.

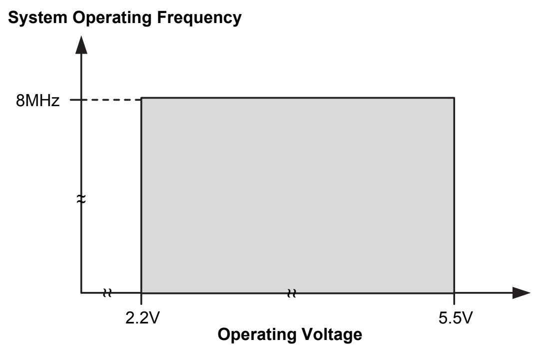
High Speed Internal Oscillator – HIRC – Frequency Accuracy

During the program writing operation the writer will trim the HIRC oscillator at a user selected HIRC frequency and user selected voltage of 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%	8	+1%	MHz
			-40°C~85°C	-2%	8	+2%	
		2.2V~5.5V	25°C	-2.5%	8	+2.5%	MHz
			-40°C~85°C	-3%	8	+3%	

Note: 1. The 3V/5V values for V_{DD} are provided as these are the two selected 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 2.2V to 3.6V and fixed at 5V for application voltage ranges from 3.3V to 5.5V.

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	—	$f_{SYS}=f_H \sim f_H/64$, $f_H=f_{HIRC}$	—	16	—	t_{HIRC}
	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_H
t_{RSTD}	System Reset Delay Time Reset Source from Power-on Reset or LVR Reset	—	RR_{POR}=5V/ms	42	48	54	ms
	System Reset Delay Time Reset Source from LVRC/WDTC Software Reset	—	—				
	System Reset Delay Time Reset Source from WDT Overflow	—	—	14	16	18	ms
t_{SRESET}	Minimum Software Reset Width to Reset	—	—	45	90	120	μs

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 $t_{HIRC}=1/f_{HIRC}$, etc.

Input/Output Characteristics

Ta=-40°C~85°C

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	—	$0.2V_{DD}$	
V_{IH}	Input High Voltage for I/O Ports	5V	—	3.5	—	5.0	V
		—	—	$0.8V_{DD}$	—	V_{DD}	
I_{OL}	Sink Current for I/O Ports	3V	$V_{OL}=0.1V_{DD}$	16	32	—	mA
		5V		32	65	—	
I_{OH}	Source Current for I/O Ports	3V	$V_{OH}=0.9V_{DD}$	-4	-8	—	mA
		5V		-8	-16	—	
R_{PH}	Pull-high Resistance for I/O Ports (Note)	3V	—	20	60	100	kΩ
		5V		10	30	50	
I_{LEAK}	Input Leakage Current	5V	$V_{IN}=V_{DD}$ or $V_{IN}=V_{SS}$	—	—	±1	μA
t_{TCK}	CTMn/STM External Clock Input Minimum Pulse Width	—	—	0.3	—	—	μs
t_{INT}	External Interrupt Input Minimum Pulse Width	—	—	10	—	—	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				
Flash Program / Emulated EEPROM Memory							
V_{DD}	V_{DD} for Read	—	—	2.2	—	5.5	V
	V_{DD} for Erase/Write			2.2	—	5.5	V
t_{DEW}	Erase/Write Cycle Time – Flash Program Memory	5V	—	—	2	3	ms
	Erase / Write Cycle Time – Emulated EEPROM Memory	—	EWRTS[1:0]=00B	—	2	3	ms
			EWRTS[1:0]=01B	—	4	6	
			EWRTS[1:0]=10B	—	8	12	
			EWRTS[1:0]=11B	—	16	24	
I_{DDPGM}	Programming/Erase current on V_{DD}	—	—	—	—	5	mA
E_P	Cell Endurance	—	—	10K	—	—	E/W
t_{RETD}	ROM Data Retention Time	—	Ta=25°C	—	40	—	Year
RAM Data Memory							
V_{DD}	Operating Voltage for Read/Write	—	—	V_{DDmin}	—	V_{DDmax}	V
V_{DR}	RAM Data Retention Voltage	—	Device in SLEEP Mode	1.0	—	—	V

Note: The Emulated EEPROM erase/write operation can only be executed when the f_{SYS} clock frequency is equal to or greater than 2MHz.

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 2.1V	-5%	2.10	+5%	V
		—	LVR enable, voltage select 2.55V		2.55		
		—	LVR enable, voltage select 3.15V		3.15		
		—	LVR enable, voltage select 3.8V		3.80		
I_{LVRBG}	Operating Current	3V	LVR enable	—	—	18	μA
		5V		—	20	25	
I_{LVR}	Additional Current for LVR Enable	—	—	—	—	24	μA
t_{LVR}	Minimum Low Voltage Width to Reset	—	—	120	240	480	μs

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	—	AV_{DD}	V
N_R	Resolution	—	—	—	—	10	Bit
DNL	Differential Non-linearity	—	$V_{REF}=AV_{DD}$, $t_{ADCK}=0.5\mu s$	-1.5	—	+1.5	LSB
INL	Integral Non-linearity	—	$V_{REF}=AV_{DD}$, $t_{ADCK}=0.5\mu s$	-2	—	+2	LSB
I_{ADC}	Additional Current for A/D Converter Enable	5V	No load, $t_{ADCK}=0.5\mu s$	—	300	420	μA
t_{ADCK}	A/D Conversion Clock Period	—	—	0.5	—	10.0	μs
t_{ON2ST}	A/D Converter On-to-Start Time	—	—	4	—	—	μs
t_{ADC}	Conversion Time (Including A/D Sample and Hold Time)	—	—	—	14	—	t_{ADCK}

LCD Driver Electrical Characteristics

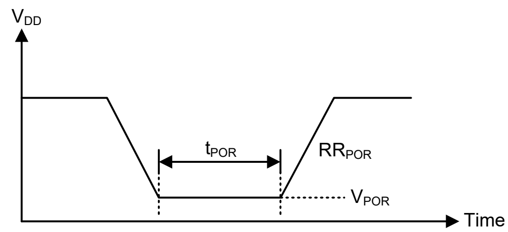
Ta=-40°C~85°C

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
I_{BIAS}	LCD Bias Current	5V	ISEL[2:0]=000B	5.81	8.30	10.79	μA
			ISEL[2:0]=001B	11.62	16.60	21.58	
			ISEL[2:0]=010B	35	50	65	
			ISEL[2:0]=011B	70	100	130	
			ISEL[2:0]=100B	350	500	650	
			ISEL[2:0]=101B	700	1000	1300	
			ISEL[2:0]=110B	1400	2000	2600	
			ISEL[2:0]=111B	2800	4000	5200	
V_{SCOM}	$2/3V_{DD}$ Voltage for LCD SCOM Output	2.2V~5.5V	No load	0.63	0.66	0.69	V_{DD}
	$1/3V_{DD}$ Voltage for LCD SCOM Output	2.2V~5.5V	No load	0.31	0.33	0.35	

Power-on Reset Characteristics

Ta=25°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

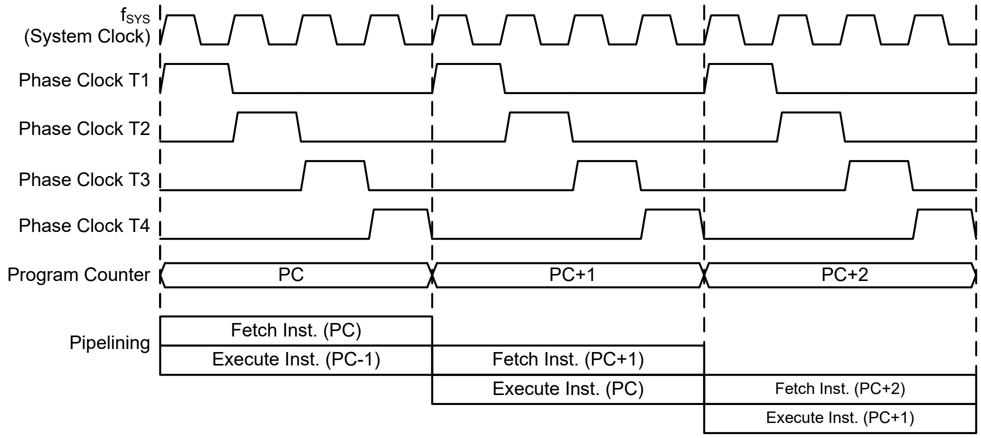


System Architecture

A key factor in the high-performance features of the Holtek range of 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 and A/D control system with maximum reliability and flexibility. This makes the device suitable for low-cost, high-volume production for controller applications.

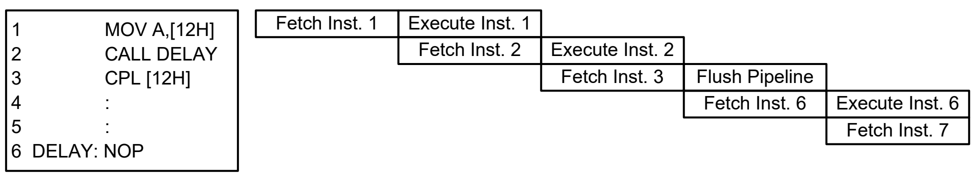
Clocking and Pipelining

The main system clock, derived from HIRC 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.



System Clocking and Pipelining

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.



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 demands 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	
High Byte	**Low Byte (PCL Register)**
PC11~PC8	PCL7~PCL0

Program Counter

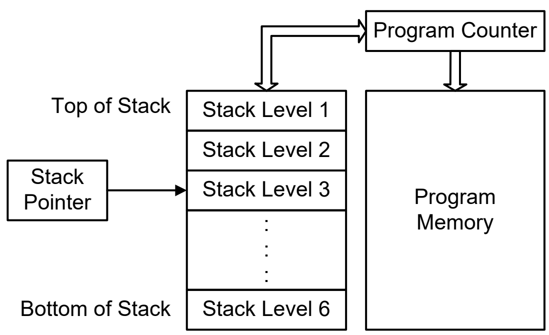
The lower byte of the Program Counter, known as the Program Counter Low register or PCL, is available for program control and is a readable and writeable register. By transferring data directly into this register, a short program jump can be executed directly. However, as only this low byte is available 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 is organized into 6 levels and 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:

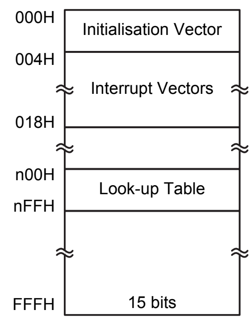
- Arithmetic operations: ADD, ADDM, ADC, ADCM, SUB, SUBM, SBC, SBCM, DAA
- Logic operations: AND, OR, XOR, ANDM, ORM, XORM, CPL, CPLA
- Rotation: RRA, RR, RRCA, RRC, RLA, RL, RLCA, RLC
- Increment and Decrement: INCA, INC, DECA, DEC
- Branch decision: JMP, SZ, SZA, SNZ, SIZ, SDZ, SIZA, SDZA, CALL, RET,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

Flash Program Memory

The Program Memory is the location where the user code or program is stored. For this device the Program Memory is Flash type, which means it can be programmed and re-programmed a large number of times, allowing the user the convenience of code modification on the same device. By using the appropriate programming tools, the Flash device offer users the flexibility to conveniently debug and develop their applications while also offering a means of field programming and updating.

Structure

The Program Memory has a capacity of 4K×15 bits. The Program Memory is addressed by the Program Counter and also contains data, table information and interrupt entries. Table data, which can be arranged 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 registers, TBLP and TBHP. These registers define 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 bit in this transferred higher order byte will be read as "0".

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

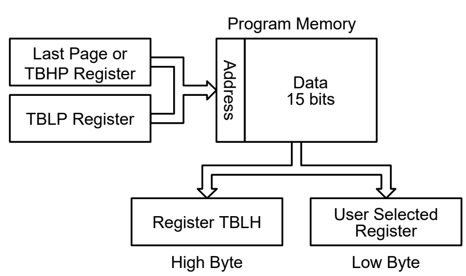


Table Program Example

The following example shows how the table pointer and table data is defined and retrieved from the microcontroller. 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 “F00H” which refers to the start address of the last page within the 4K words Program Memory of the device. The table pointer low byte register is setup 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 “F06H” or 6 locations after the start of the last page. Note that the value for the table pointer is referenced to the specific address pointed by the TBLP and TBHP registers 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 the page that tbhp pointed
mov a,0Fh       ; initialise high table pointer
mov tbhp,a
:
:
tabrd tempreg1  ; transfers value in table referenced by table pointer,
                ; data at program memory address "F06H" 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 "F05H" 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 F00h        ; sets initial address of program memory
dc 00Ah, 00Bh, 00Ch, 00Dh, 00Eh, 00Fh, 01Ah, 01Bh
:
```

In Circuit Programming – ICP

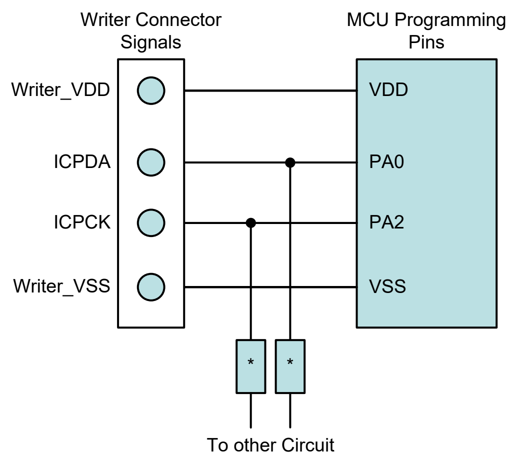
The provision of Flash type Program Memory provides the user with a means of convenient and easy upgrades and modifications to their programs on the same device.

As an additional convenience, Holtek has provided a means of programming the microcontroller in-circuit using a 4-pin interface. This provides manufacturers with the possibility of manufacturing their circuit boards complete with a programmed or un-programmed microcontroller, and then programming or upgrading the program at a later stage. This enables product manufacturers to easily keep their manufactured products supplied with the latest program releases without removal and re-insertion of the device.

Holtek Writer Pins	MCU Programming Pins	Pin Description
ICPDA	PA0	Programming Serial Data/Address
ICPCK	PA2	Programming Clock
VDD	VDD	Power Supply
VSS	VSS	Ground

The Program Memory can be programmed serially in-circuit using this 4-wire interface. Data is downloaded and uploaded serially on a single pin with an additional line for the clock. Two additional lines are required for the power supply. The technical details regarding the in-circuit programming of the device are beyond the scope of this document and will be supplied in supplementary literature.

During the programming process, the user can 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: * 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 HT66V0184 which is used to emulate the HT66F0184 device. 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 Holtek HT-IDE 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. For more detailed OCDS information, refer to the corresponding document named "Holtek e-Link for 8-bit MCU OCDS User's Guide".

Holtek e-Link 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

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.

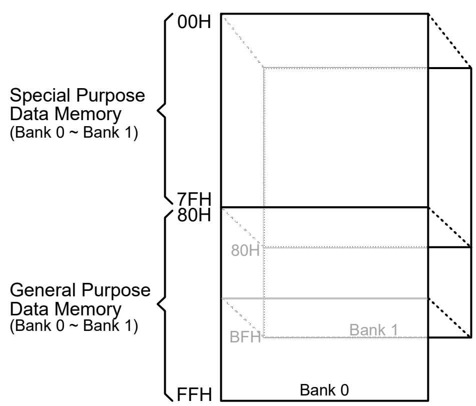
Structure

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 overall Data Memory is subdivided into two banks, which are implemented in 8-bit wide Memory. The Special Purpose Data Memory registers are accessible in all banks. The address range of the Special Purpose Data Memory for the device is from 00H to 7FH while the address range of the General Purpose Data Memory is from 80H to FFH in Bank0 and from 80H to BFH in Bank1. Switching between the different Data Memory banks is achieved by properly setting the Bank Pointer to the correct value.

Special Purpose Data Memory	General Purpose Data Memory	
Located Bank	Capacity	Bank: Address
0~1	192×8	0: 80H~FFH 1: 80H~BFH

Data Memory Summary



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 programing 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".

Address	Bank 0~1
00H	IAR0
01H	MP0
02H	IAR1
03H	MP1
04H	BP
05H	ACC
06H	PCL
07H	TBLP
08H	TBLH
09H	TBHP
0AH	STATUS
0BH	(Unused)
0CH	(Unused)
0DH	(Unused)
0EH	(Unused)
0FH	RSTFC
10H	(Unused)
11H	(Unused)
12H	(Unused)
13H	(Unused)
14H	PA
15H	PAC
16H	PAPU
17H	PAWU
18H	SCC
19H	INTEG
1AH	INTC0
1BH	INTC1
1CH	MFI0
1DH	MFI1
1EH	WDTC
1FH	PB
20H	PBC
21H	PBPU
22H	PC
23H	PCC
24H	PCPU
25H	PD
26H	PDC
27H	PDPU
28H	SADOL
29H	SADOH
2AH	SADC0
2BH	SADC1
2CH	STMC0
2DH	STMC1
2EH	STMDL
2FH	STMDH
30H	STMAL
31H	STMAH
32H	CTM1C0
33H	CTM1C1
34H	CTM1DL
35H	CTM1DH
36H	CTM1AL
37H	CTM1AH
38H	CTM0C0
39H	CTM0C1
3AH	CTM0DL
3BH	CTM0DH
3CH	CTM0AL
3DH	CTM0AH
3EH	EAR
3FH	ECR

Address	Bank 0~1
40H	ED0L
41H	ED0H
42H	ED1L
43H	ED1H
44H	ED2L
45H	ED2H
46H	ED3L
47H	ED3H
48H	SLCDC0
49H	SLCDC1
4AH	SLCDC2
4BH	SLCDC3
4CH	SLCDC4
4DH	SLCDC5
4EH	LVRC
4FH	IFS
50H	PASR
51H	PDSR
52H ~ 7FH	(Unused)

(Unused): Unused, read as 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 their locations in normal RAM register space, do not actually physically exist as normal registers. The method of indirect addressing for RAM data manipulation uses these 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 their corresponding Memory Pointers, MP0 or MP1. Acting as a pair, IAR0 and MP0 can together access data from Bank 0 while the IAR1 register together with the MP1 register can access data from any Data Memory Bank. As the Indirect Addressing Registers are not physically implemented, reading the Indirect Addressing Registers will return a result of "00H" and writing to the registers 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 relevant Indirect Addressing Registers is carried out, the actual address that the microcontroller is directed to is the address specified by the related Memory Pointer. MP0, together with Indirect Addressing Register, IAR0, are used to access data from Bank 0, while MP1 together with IAR1 are used to access data from all data banks according to the BP register. Direct Addressing can only be used with Bank 0, all other Banks must be addressed indirectly using MP1 and IAR1.

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              ; increment 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 example shown above, no reference is made to specific Data Memory addresses.

Bank Pointer – BP

The Data Memory is divided into two banks, Banks 0~1. Selecting the required Data Memory bank is achieved using the Bank Pointer. Bits 0 of the Bank Pointer is used to select Data Memory Banks 0~1. The Data Memory is initialised to Bank 0 after a reset, except for a WDT time-out reset in the IDLE or SLEEP Mode, in which case, the selected Data Memory bank remains unaffected. It should be noted that the Special Function Data Memory is not affected by the bank selection, which means that the Special Function Registers can be accessed within any bank. Directly addressing the Data Memory will always result in Bank 0 being accessed irrespective of the value of the Bank Pointer. Accessing data from banks other than Bank 0 must be implemented using the indirect addressing.

- **BP Register**

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	—	—	—	DMBP0
R/W	—	—	—	—	—	—	—	R/W
POR	—	—	—	—	—	—	—	0

Bit 7~1 Unimplemented, read as “0”

Bit 0 **DMBP0**: Data Memory Bank selection
0: Bank 0
1: Bank 1

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 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, TBHP, TBLH

These three special function registers are used to control operation of the look-up table which is stored in the Program Memory. TBLP and TBHP are the table pointers and indicate the location where the table data is located. Their value must be set before any table read commands are executed. Their 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.

Emulated EEPROM Data Memory

The device contains an Emulated EEPROM Data Memory, which is by its nature a non-volatile form of re-programmable memory, with data retention even when its power supply is removed. By incorporating this kind of data memory, a whole new host of application possibilities are made available to the designer. The availability of the Emulated EEPROM storage allows information such as product identification numbers, calibration values, specific user data, system setup data or other product information to be stored directly within the product microcontroller.

Emulated EEPROM Data Memory Structure

The Emulated EEPROM Data Memory capacity is 32×15 bits for the device. The Emulated EEPROM Erase operation is carried out in a page format while the Write operation is carried out in 4-word format and Read operation in a word format. The page size is assigned with a capacity of 16 words. Note that the Erase operation should be executed before the Write operation is executed.

Operations	Format
Erase	1 page/time
Write	4 words/time
Read	1 word/time
Note: Page size=16 words	

Emulated EEPROM Erase/Write/Read Format

Erase Page	EAR4	EAR [3:0]
0	0	xxxx
1	1	xxxx

"x": don't care

Erase Page Number and Selection

Write Unit	EAR[4:2]	EAR[1:0]
0	000	xx
1	001	xx
2	010	xx
3	011	xx
4	100	xx
5	101	xx
6	110	xx
7	111	xx

"x": don't care

Write Unit Number and Selection

Emulated EEPROM Registers

Several registers control the overall operation of the Emulated EEPROM Data Memory. These are the address register, EAR, the data registers, ED0L&ED0H ~ ED3L&ED3H, and a single control register, ECR.

Register Name	Bit							
	7	6	5	4	3	2	1	0
EAR	—	—	—	EAR4	EAR3	EAR2	EAR1	EAR0
ED0L	D7	D6	D5	D4	D3	D2	D1	D0
ED0H	—	D14	D13	D12	D11	D10	D9	D8
ED1L	D7	D6	D5	D4	D3	D2	D1	D0
ED1H	—	D14	D13	D12	D11	D10	D9	D8
ED2L	D7	D6	D5	D4	D3	D2	D1	D0
ED2H	—	D14	D13	D12	D11	D10	D9	D8
ED3L	D7	D6	D5	D4	D3	D2	D1	D0
ED3H	—	D14	D13	D12	D11	D10	D9	D8
ECR	EWRTS1	EWRTS0	EEREN	EER	EWREN	EWR	ERDEN	ERD

Emulated EEPROM Register List

- **EAR Register**

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	EAR4	EAR3	EAR2	EAR1	EAR0
R/W	—	—	—	R/W	R/W	R/W	R/W	R/W
POR	—	—	—	0	0	0	0	0

Bit 7~5 Unimplemented, read as "0"

Bit 4~0 **EAR4~EAR0**: Emulated EEPROM address bit 4 ~ bit 0

- **ED0L 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**: The first Emulated EEPROM data bit 7 ~ bit 0

- **ED0H 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**: The first Emulated EEPROM data bit 14 ~ bit 8

- **ED1L 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**: The second Emulated EEPROM data bit 7 ~ bit 0

• **ED1H 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**: The second Emulated EEPROM data bit 14 ~ bit 8

• **ED2L 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**: The third Emulated EEPROM data bit 7 ~ bit 0

• **ED2H 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**: The third Emulated EEPROM data bit 14 ~ bit 8

• **ED3L 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**: The fourth Emulated EEPROM data bit 7 ~ bit 0

• **ED3H 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**: The fourth Emulated EEPROM data bit 14 ~ bit 8

• **ECR Register**

Bit	7	6	5	4	3	2	1	0
Name	EWRTS1	EWRTS0	EEREN	EER	EWREN	EWR	ERDEN	ERD
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 **EWRTS1~EWRTS0**: Emulated EEPROM Erase/Write cycle time selection
00: 2ms
01: 4ms
10: 8ms
11: 16ms

Bit 5 **EEREN**: Emulated EEPROM Erase enable
0: Disable
1: Enable
This bit is used to enable the Emulated EEPROM erase function and must be set high before erase operations are carried out. This bit will be automatically reset to zero by the hardware after the erase cycle has finished. Clearing this bit to zero will inhibit the Emulated EEPROM erase operations.

Bit 4 **EER**: Emulated EEPROM Erase control
0: Erase cycle has finished
1: Activate an erase cycle
When this bit is set high by the application program, an erase cycle will be activated. This bit will be automatically reset to zero by the hardware after the erase cycle has finished. Setting this bit high will have no effect if the EEREN has not first been set high.

Bit 3 **EWREN**: Emulated EEPROM Write enable
0: Disable
1: Enable
This bit is used to enable the Emulated EEPROM write function and 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 the Emulated EEPROM write operations.

Bit 2 **EWR**: Emulated EEPROM Write control
0: Write cycle has finished
1: Activate a write cycle
When this bit is set high by the application program, a write cycle will be activated. 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 EWREN has not first been set high.

Bit 1 **ERDEN**: Emulated EEPROM Read enable
0: Disable
1: Enable
This bit is used to enable the Emulated EEPROM read function and must be set high before read operations are carried out. Clearing this bit to zero will inhibit the Emulated EEPROM read operations.

Bit 0 **ERD**: Emulated EEPROM Read control
0: Read cycle has finished
1: Activate a read cycle
When this bit is set high by the application program, a read cycle will be activated. This bit will be automatically reset to zero by the hardware after the read cycle has finished. Setting this bit high will have no effect if the ERDEN has not first been set high.

Note: 1. The EEREN, EER, EWREN, EWR, ERDEN and ERD cannot be set to "1" at the same time in one instruction.
2. Note that the CPU will be stopped when a read, write or erase operation is successfully activated.
3. Ensure that the f_{SYS} clock frequency is equal to or greater than 2MHz and the f_{SUB} clock is stable before executing the erase or write operation.
4. Ensure that the read, write or erase operation is totally complete before executing other operations.

Erasing the Emulated EEPROM

For Emulated EEPROM erase operation the desired erase page address should first be placed in the EAR register. The number of the erase operation is 16 words per page each time, therefore, the available page erase address is only specified by the EAR4 bit in the EAR register and the content of EAR3~EAR0 in the EAR register is not used to specify the page address. To erase the Emulated EEPROM page, the EEREN bit in the ECR register must first be set high to enable the erase function. After this the EER bit in the ECR register must be immediately set high to initiate an erase cycle. These two instructions must be executed in two consecutive instruction cycles to activate an erase operation successfully. The global interrupt bit EMI should also first be cleared before implementing any erase operations, and then set high again after the a valid erase activation procedure has completed. Note that the CPU will be stopped when an erase operation is successfully activated. When the erase cycle terminates, the CPU will resume executing the application program. And the EER bit will be automatically cleared to zero by the microcontroller, informing the user that the data has been erased. The Emulated EEPROM erased page content will all be zero after an erase operation.

Writing Data to the Emulated EEPROM

For Emulated EEPROM write operation, the data should first be placed in the ED0L/ED0H ~ ED3L/ED3H registers, and the desired write unit address in the EAR register. The number of the write operation is 4 words each time, therefore, the available write unit address is only specified by the EAR4~EAR2 bits in the EAR register and the content of EAR1~EAR0 in the EAR register is not used to specify the unit address. To write data to the Emulated EEPROM, the EWREN bit in the ECR register must first be set high to enable the write function. After this the EWR bit in the ECR 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 high 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 EWR bit will be automatically cleared to zero by the microcontroller, informing the user that the data has been written to the Emulated EEPROM.

Reading Data from the Emulated EEPROM

For Emulated EEPROM read operation the desired read address should first be placed in the EAR register. To read data from the Emulated EEPROM, the ERDEN bit in the ECR register must first be set high to enable the read function. After this a read cycle will be initiated if the ERD bit in the ECR register is now set high. Note that the CPU will be stopped when the read operation is successfully activated. When the read cycle terminates, the CPU will resume executing the application program. And the ERD bit will be automatically cleared to zero by the microcontroller, informing the user that the data has been read from the Emulated EEPROM. Then the data can be read from the ED0H/ED0L data register pair by application program. The data will remain in the data register pair until another read, write or erase operation is executed.

Programming Considerations

Care must be taken that data is not inadvertently written to the Emulated EEPROM. 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 or erasing data the EWR or EER bit must be set high immediately after the EWREN or EEREN bit has been set high, to ensure the write or erase cycle executes correctly. The global interrupt bit EMI should also be cleared before a write or erase cycle is executed and then set again after a valid write or erase activation procedure has completed. Note that the device should not enter the IDLE or SLEEP mode until Emulated EEPROM read, write or erase operation is totally complete. Otherwise, Emulated EEPROM read, write or erase operation will fail.

Programming Examples

Erase a Data Page of the Emulated EEPROM – polling method

```
MOV  A, EEPROM_ADRES      ; user-defined page
MOV  EAR, A
MOV  A, 00H               ; Erase time=2ms (40H for 4ms, 80H for 8ms, C0H for 16ms)
MOV  ECR, A
CLR  EMI
SET  EEREN                ; set EEREN bit, enable erase operation
SET  EER                  ; start Erase Cycle - set EER bit - executed immediately after
                          ; setting EEREN bit
SET  EMI
BACK:
SZ   EER                  ; check for erase cycle end
JMP  BACK
:
```

Writing Data to the Emulated EEPROM – polling method

```
MOV  A, EEPROM_ADRES      ; user-defined address
MOV  EAR, A
MOV  A, EEPROM_DATA0_L    ; user-defined data
MOV  ED0L, A
MOV  A, EEPROM_DATA0_H
MOV  ED0H, A
MOV  A, EEPROM_DATA1_L
MOV  ED1L, A
MOV  A, EEPROM_DATA1_H
MOV  ED1H, A
MOV  A, EEPROM_DATA2_L
MOV  ED2L, A
MOV  A, EEPROM_DATA2_H
MOV  ED2H, A
MOV  A, EEPROM_DATA3_L
MOV  ED3L, A
MOV  A, EEPROM_DATA3_H
MOV  ED3H, A
MOV  A, 00H             ; Write time=2ms (40H for 4ms, 80H for 8ms, C0H for 16ms)
MOV  ECR, A
CLR  EMI
SET  EWREN              ; set EWREN bit, enable write operation
SET  EWR                ; start Write Cycle - set EWR bit - executed immediately after
                        ; setting EWREN bit
SET  EMI
BACK:
SZ   EWR                ; check for write cycle end
JMP  BACK
:
```

Reading Data from the Emulated EEPROM – polling method

```
MOV  A, EEPROM_ADRES    ; user defined address
MOV  EAR, A
SET  ERDEN              ; set ERDEN bit, enable read operation
SET  ERD                ; start Read Cycle - set ERD bit
BACK:
SZ   ERD                ; check for read cycle end
JMP  BACK
CLR  ECR                ; disable Emulated EEPROM read if no more read operations are
                        ; required
MOV  A, ED0L            ; move read data which is placed in the ED0L/ED0H to user-defined
                        ; registers
```

```
MOV  READ_DATA_L, A
MOV  A, ED0H
MOV  READ_DATA_H, A
```

Note: For each read operation, the address register should be re-specified followed by setting the ERD bit high to activate a read cycle even if the target address is consecutive.

Oscillator Configuration

The internal RC oscillator is a fully integrated system oscillator requiring no external components. The internal RC oscillator has a 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.

Operating Modes and System Clocks

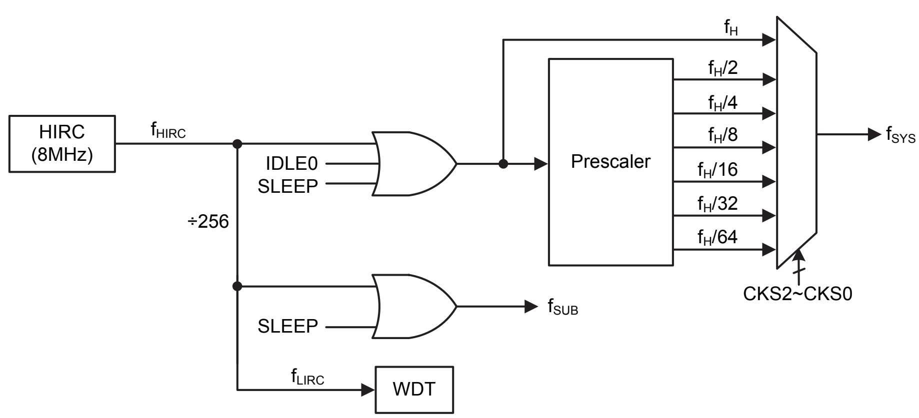
Present day applications require that their microcontrollers have high performance but often still demand that they consume as little power as possible, conflicting requirements that are especially true in battery powered portable applications. The fast clocks required for high performance will by their nature increase current consumption and of course vice-versa, lower speed clocks reduce current consumption. As Holtek has provided the device with different system clock frequencies 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 one clock source 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 comes from the high frequency f_H or a divided version of the f_H which is selected using the CKS2~CKS0 bits in the SCC register. The system clock is sourced from the HIRC oscillator.

There are two additional internal clocks for the peripheral circuits, the f_{SUB} and f_{LIRC}. Each of these internal clocks which are from a divided output of the HIRC oscillator, has an approximate frequency of 32kHz.



Device Clock Configurations

System Operation Modes

There are four different modes of operation for the microcontroller, each one with its own special characteristics and which can be chosen according to the specific performance and power requirements of the application. There is one mode allowing normal operation of the microcontroller, the FAST Mode. The remaining three modes, the SLEEP, IDLE0 and IDLE1 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_{HIRC}	f_{LIRC}
		IDLEEN	FHIDEN	CKS2~CKS0					
FAST	On	x	x	000~110	f_H~f_H/64	On	On	On	On
IDLE1	Off	1	1	000~110	On	On	On	On	On
IDLE0	Off	1	0	xxx	Off	Off	On	On	On
SLEEP	Off	0	x	xxx	Off	Off	Off	On/Off (note)	On/Off(note)

"x": Don't care

Note: In the SLEEP mode, if the WDT function is enabled, the f_{HIRC} and f_{LIRC} will be on; otherwise the clocks will be off.

FAST Mode

In this mode the microcontroller has all of its functions operational and where the system clock is provided by the high speed oscillator. This mode operates allowing the microcontroller to operate normally with a clock source which will come from HIRC oscillator. The HIRC 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 microcontroller at a divided clock ratio reduces the operating current.

SLEEP Mode

The SLEEP Mode is entered when a HALT instruction is executed and when the IDLEEN bit in the SCC register is low. In the SLEEP mode the CPU will be stopped. The f_{SUB} clock provided to the peripheral functions will also be stopped, too, however the Watchdog Timer function can be on or off which is decided by user application.

IDLE0 Mode

The IDLE0 Mode is entered when a HALT instruction is executed and when the IDLEEN bit in the SCC register is high and the FHIDEN bit in the SCC register is low. In the IDLE0 Mode the system oscillator will be off and inhibited from driving the CPU but the f_{SUB} clock to peripheral will be continue to provide a clock source to keep some peripheral functions operational.

IDLE1 Mode

The IDLE1 Mode is entered when a HALT instruction is executed and when the IDLEEN bit in the SCC register is high and the FHIDEN bit in the SCC register is high. In the IDLE1 Mode the system oscillator will be inhibited from driving the CPU but may continue to provide a clock source to keep some peripheral functions operational.

Control Register

The SCC register is used to control the internal clocks within the device.

- **SCC Register**

Bit	7	6	5	4	3	2	1	0
Name	CKS2	CKS1	CKS0	—	—	—	FHIDEN	IDLEEN
R/W	R/W	R/W	R/W	—	—	—	R/W	R/W
POR	0	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: Reserved
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, a divided version of the high speed system oscillator can also be chosen as the system clock source.

Bit 4~2 Unimplemented, read as "0"

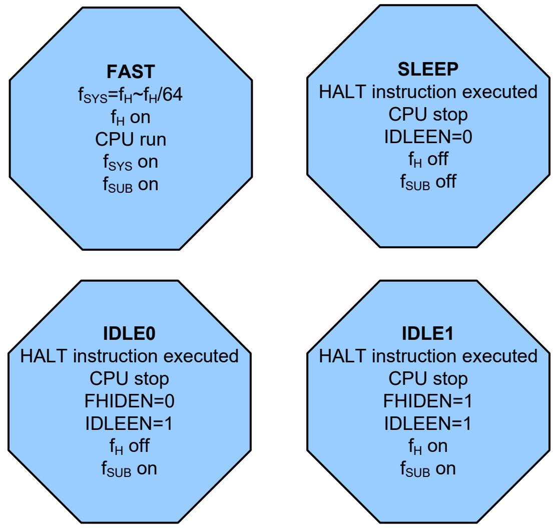
Bit 1 **FHIDEN**: High frequency clock f_H control in IDLE mode
0: Disable
1: Enable
This bit is used to determine the f_H clock is on or off in the IDLE mode. If this bit is set high and the IDLEEN bit is high, the f_H clock will continue to run and the device will enter the IDLE1 mode after a HALT instruction is executed. However, the f_H clock will be stopped and the device will enter the IDLE0 mode after a HALT instruction is executed if this bit is low with the IDLEEN bit set high.

Bit 0 **IDLEEN**: IDLE mode control
0: Disable – SLEEP mode
1: Enable – IDLE mode
This bit is used to determine what happens when the HALT instruction is executed. If this bit is high, the device will enter the IDLE mode when a HALT instruction is executed. Otherwise, the device will enter the SLEEP mode when a HALT instruction is executed if this bit is set low.

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 from the FAST Mode to the SLEEP/IDLE Mode is executed via the HALT instruction. When a HALT instruction is executed, whether the device enters the IDLE Mode or the SLEEP Mode is determined by the condition of the IDLEEN bit in the SCC register.



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 the IDLEEN bit in 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 is disabled then the WDT will be cleared and 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 IDLEEN bit in SCC register equal to "1" and the FHIDEN bit in 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 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 is disabled then the WDT will be cleared and 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 the IDLEEN bit in SCC register equal to "1" and the FHIDEN bit in 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 is disabled then the WDT will be cleared and 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 IDLE 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.

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 f_{LIRC} has enabled.

Wake-up

To minimise power consumption the device can enter the SLEEP or IDLE 0~1 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
- 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 a WDT overflow, a Watchdog Timer reset will be initiated and the TO flag will be set to 1. 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 a divided version with a division ratio of 256 of the HIRC oscillator. The f_{LIRC} clock has an approximate frequency of 32kHz. The Watchdog Timer source clock is then subdivided by a ratio of 2^8 to 2^{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	WE4	WE3	WE2	WE1	WE0	WS2	WS1	WS0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	1	0	1	0	1	1	1

Bit 7~3 **WE4~WE0**: WDT function software control
01010: Enable
10101: Disable
Other values: Reset MCU
When these bits are changed to any other values due to environmental noise the microcontroller will be reset; this reset operation will be activated after a delay time, t_{SRESET}, and the WRF bit in the RSTFC register will be set high.

Bit 2~0 **WS2~WS0**: WDT time-out period selection
000: $[(2^8-2^0)\sim2^8]/f_{LIRC}$
001: $[(2^9-2^1)\sim2^9]/f_{LIRC}$
010: $[(2^{10}-2^2)\sim2^{10}]/f_{LIRC}$
011: $[(2^{11}-2^3)\sim2^{11}]/f_{LIRC}$
100: $[(2^{12}-2^4)\sim2^{12}]/f_{LIRC}$
101: $[(2^{13}-2^5)\sim2^{13}]/f_{LIRC}$
110: $[(2^{14}-2^6)\sim2^{14}]/f_{LIRC}$
111: $[(2^{15}-2^7)\sim2^{15}]/f_{LIRC}$
These three bits determine the division ratio of the Watchdog Timer source clock, which in turn determines the timeout period. The f_{LIRC} clock is a divided version with a division ratio of 256 of the HIRC oscillator and has an approximate frequency of 32kHz.

• **RSTFC Register**

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	—	LVRF	LRF	WRF
R/W	—	—	—	—	—	R/W	R/W	R/W
POR	—	—	—	—	—	x	0	0

Bit 7~3 Unimplemented, read as "0"

Bit 2 **LVRF**: LVR function reset flag
Refer to the Low Voltage Reset section.

Bit 1 **LRF**: LVR control register software reset flag
Refer to the Low Voltage Reset section.

Bit 0 **WRF**: WDT control register software reset flag
0: Not occurred
1: Occurred
This bit is set to 1 by the WDT Control register software reset and cleared to 0 by the application program. Note that this bit can only be cleared to 0 by the application program.

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. There are five bits, WE4~WE0, in the WDTC register to offer the enable/disable control and reset control of the Watchdog Timer. The WDT function will be disabled when the WE4~WE0 bits are set to a value of 10101B while the WDT function will be enabled if the WE4~WE0 bits are equal to 01010B. If the WE4~WE0 bits are set to any other values, other than 01010B and 10101B, it will reset the device after a delay time, t_{SRESET}. After power on these bits will have a value of 01010B.

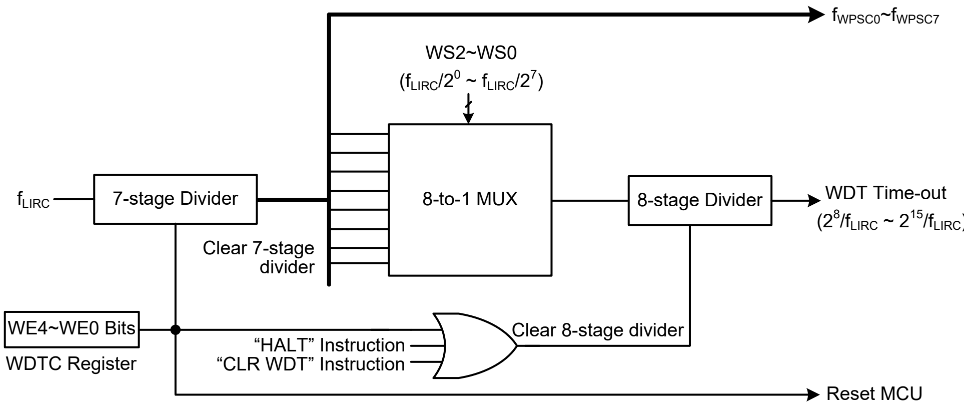
WE4~WE0 Bits	WDT Function
10101B	Disable
01010B	Enable
Any other value	Reset MCU

Watchdog Timer 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. Three methods can be adopted to clear the contents of the Watchdog Timer. The first is a WDTC software reset, which means a certain value except 01010B and 10101B written into the WE4~WE0 bits, the second is using the Watchdog Timer software clear instruction, the third is via a HALT instruction.

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 clock 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 microcontroller. In this case, internal circuitry will ensure that the microcontroller, 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.

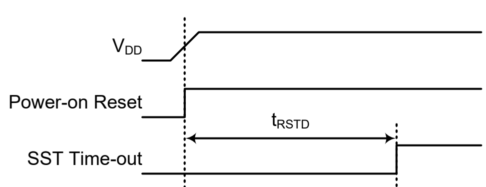
Another reset exists in the form of a Low Voltage Reset, LVR, where a full reset is implemented in situations where the power supply voltage falls below a certain threshold. Another type of reset is when the Watchdog Timer overflows and resets the microcontroller. All types of reset operations result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

Reset Functions

There are several ways in which a microcontroller reset can occur, each of which will be described as follows.

Power-on Reset

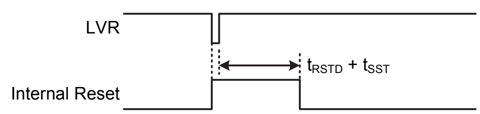
The most fundamental and unavoidable reset is the one that occurs after power is first applied to the microcontroller. As well as ensuring that the Program Memory begins execution from the first memory address, a power-on reset also ensures that certain other registers are preset to known conditions. 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

Low Voltage Reset – LVR

The microcontroller contains a low voltage reset circuit in order to monitor the supply voltage of the device and provides an MCU reset should the value fall below a certain predefined level. 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 and the LVRF bit in the RSTFC register will also be set high. 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 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 actual V_{LVR} value can be selected by the LVS7~LVS0 bits in the LVRC register. If the LVS7~LVS0 bits are changed to some different values by environmental noise, the LVR will reset the device after a delay time, t_{SRESET}. When this happens, the LRF bit in the RSTFC register will be set high. After power on the register will have the value of 01010101B. Note that the LVR function will be automatically disabled when the device enters the IDLE or SLEEP mode.



Low Voltage Reset Timing Chart

- **LVRC Register**

Bit	7	6	5	4	3	2	1	0
Name	LVS7	LVS6	LVS5	LVS4	LVS3	LVS2	LVS1	LVS0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	1	0	1	0	1	0	1

Bit 7~0 **LVS7~LVS0**: LVR voltage select control
01010101: 2.1V
00110011: 2.55V
10011001: 3.15V
10101010: 3.8V
Any other value: Generates MCU reset – register is reset to POR value

When an actual low voltage condition occurs, as specified above, an MCU reset will be generated. The reset operation will be activated after the low voltage condition keeps more than a t_{LVR} time. In this situation the register contents will remain the same after such a reset occurs.

Any register value, other than the four defined register value above, will also result in the generation of an MCU reset. The reset operation will be activated after a delay time, t_{SRESET}. However in this situation the register contents will be reset to the POR value.

- **RSTFC Register**

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	—	LVRF	LRF	WRF
R/W	—	—	—	—	—	R/W	R/W	R/W
POR	—	—	—	—	—	x	0	0

"x": unknown

Bit 7~3 Unimplemented, read as "0"

Bit 2 **LVRF**: LVR function reset flag
0: Not occurred
1: Occurred

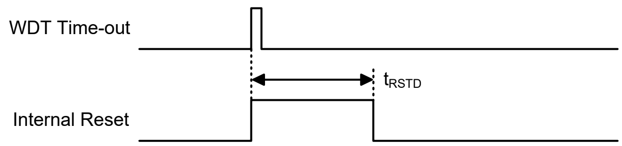
This bit is set high when a specific Low Voltage Reset situation condition occurs. This bit can only be cleared to zero by the application program.

Bit 1 **LRF**: LVR control register software reset flag
0: Not occurred
1: Occurred
This bit is set high if the LVRC register contains any non-defined LVRC register values. This in effect acts like a software-reset function. This bit can only be cleared to zero by the application program.

Bit 0 **WRF**: WDT control register software reset flag
Refer to the Watchdog Timer Control Register section.

Watchdog Time-out Reset during Normal Operation

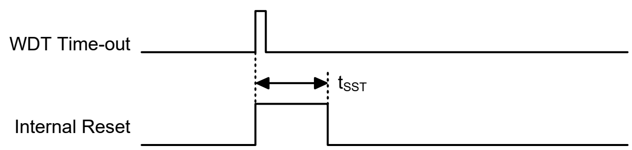
When the Watchdog time-out Reset during normal operations in the FAST mode occurs, the Watchdog time-out flag TO will be set to “1”.



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 flag 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 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	LVR reset during FAST Mode operation
1	u	WDT time-out reset during FAST 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	Cleared and then begins counting after reset
Timer Modules	Timer Module 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	Power On Reset	LVR Reset (Normal Operation)	WDT Time-out (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
BP	---- ---0	---- ---0	---- ---0	---- ---u
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
TBHP	---- xxxx	---- uuuu	---- uuuu	---- uuuu
STATUS	--00 xxxx	--uu uuuu	--1u uuuu	--11 uuuu
RSTFC	---- -x00	---- -uuu	---- -uuu	---- -uuu
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
SCC	000- --00	000- --00	000- --00	uuu- --uu
INTEG	---- --00	---- --00	---- --00	---- --uu
INTC0	-0-0 0-00	-0-0 0-00	-0-0 0-00	-u-u u-uu
INTC1	-0-0 -0-0	-0-0 -0-0	-0-0 -0-0	-u-u -u-u
MFI0	--00 --00	--00 --00	--00 --00	--uu --uu
MFI1	0000 0000	0000 0000	0000 0000	uuuu uuuu
WDTC	0101 0111	0101 0111	0101 0111	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
PC	--11 1111	--11 1111	--11 1111	--uu uuuu
PCC	--11 1111	--11 1111	--11 1111	--uu uuuu
PCPU	--00 0000	--00 0000	--00 0000	--uu uuuu
PD	--11 1111	--11 1111	--11 1111	--uu uuuu
PDC	--11 1111	--11 1111	--11 1111	--uu uuuu
PDPU	--00 0000	--00 0000	--00 0000	--uu uuuu
SADOL	xx-- ----	xx-- ----	xx-- ----	uu-- ---- (ADRFS=0)
	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu (ADRFS=1)
SADOH	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu (ADRFS=0)
	---- --xx	---- --xx	---- --xx	---- --uu (ADRFS=1)
SADC0	0000 0000	0000 0000	0000 0000	uuuu uuuu
SADC1	---- -000	---- -000	---- -000	---- -uuu
STMC0	0000 0000	0000 0000	0000 0000	uuuu uuuu
STMC1	0000 0000	0000 0000	0000 0000	uuuu uuuu

Register	Power On Reset	LVR Reset (Normal Operation)	WDT Time-out (Normal Operation)	WDT Time-out (IDLE / SLEEP)
STMDL	0000 0000	0000 0000	0000 0000	uuuu uuuu
STMDH	---- --00	---- --00	---- --00	---- --uu
STMAL	0000 0000	0000 0000	0000 0000	uuuu uuuu
STMAH	---- --00	---- --00	---- --00	---- --uu
CTM1C0	0000 0000	0000 0000	0000 0000	uuuu uuuu
CTM1C1	0000 0000	0000 0000	0000 0000	uuuu uuuu
CTM1DL	0000 0000	0000 0000	0000 0000	uuuu uuuu
CTM1DH	---- --00	---- --00	---- --00	---- --uu
CTM1AL	0000 0000	0000 0000	0000 0000	uuuu uuuu
CTM1AH	---- --00	---- --00	---- --00	---- --uu
CTM0C0	0000 0000	0000 0000	0000 0000	uuuu uuuu
CTM0C1	0000 0000	0000 0000	0000 0000	uuuu uuuu
CTM0DL	0000 0000	0000 0000	0000 0000	uuuu uuuu
CTM0DH	---- --00	---- --00	---- --00	---- --uu
CTM0AL	0000 0000	0000 0000	0000 0000	uuuu uuuu
CTM0AH	---- --00	---- --00	---- --00	---- --uu
EAR	---0 0000	---0 0000	---0 0000	---u uuuu
ECR	0000 0000	0000 0000	0000 0000	uuuu uuuu
ED0L	0000 0000	0000 0000	0000 0000	uuuu uuuu
ED0H	-000 0000	-000 0000	-000 0000	-uuu uuuu
ED1L	0000 0000	0000 0000	0000 0000	uuuu uuuu
ED1H	-000 0000	-000 0000	-000 0000	-uuu uuuu
ED2L	0000 0000	0000 0000	0000 0000	uuuu uuuu
ED2H	-000 0000	-000 0000	-000 0000	-uuu uuuu
ED3L	0000 0000	0000 0000	0000 0000	uuuu uuuu
ED3H	-000 0000	-000 0000	-000 0000	-uuu uuuu
SLCDC0	0000 ---0	0000 ---0	0000 ---0	uuuu ---u
SLCDC1	0000 0000	0000 0000	0000 0000	uuuu uuuu
SLCDC2	---- 0000	---- 0000	---- 0000	---- uuuu
SLCDC3	0000 0000	0000 0000	0000 0000	uuuu uuuu
SLCDC4	0000 0000	0000 0000	0000 0000	uuuu uuuu
SLCDC5	0000 0000	0000 0000	0000 0000	uuuu uuuu
LVRC	0101 0101	uuuu uuuu	0101 0101	uuuu uuuu
IFS	---- ---0	---- ---0	---- ---0	---- ---u
PASR	---- --00	---- --00	---- --00	---- --uu
PDSR	---0 0000	---0 0000	---0 0000	---u uuuu

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

Input/Output Ports

Holtek microcontrollers offer 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 name PA~PD. 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	—	—	PC5	PC4	PC3	PC2	PC1	PC0
PCC	—	—	PCC5	PCC4	PCC3	PCC2	PCC1	PCC0
PCPU	—	—	PCPU5	PCPU4	PCPU3	PCPU2	PCPU1	PCPU0
PD	—	—	PD5	PD4	PD3	PD2	PD1	PD0
PDC	—	—	PDC5	PDC4	PDC3	PDC2	PDC1	PDC0
PDPU	—	—	PDPU5	PDPU4	PDPU3	PDPU2	PDPU1	PDPU0

“—”: 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 an 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 PAPU~PDPU, 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 or NMOS output. 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, C or D. 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

Bit 7~0 **PAWU7~PAWU0**: PA7~PA0 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~PDC, 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 x 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, C or D. 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 devices include a Port x Output Function Selection register, labeled as PxSR, and Input Function Selection register, labeled as IFS, which can select the desired functions of the multi-function pin-shared pins. Note that the pin-shared SSEGn or SCOMm functions which have higher selection priority are selected using the SLCDC0~ SLCDC5 register that described in the Software Controlled LCD Driver section.

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 digital input pins, such as STPI and INT, which share the same pin-shared control configuration with their corresponding general purpose I/O functions when setting the relevant pin-shared control bits. To select these pin functions, in addition to the necessary pin-shared control and peripheral functional setup aforementioned, they 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
IFS	—	—	—	—	—	—	—	INTPS
PASR	—	—	—	—	—	—	PAS1	PAS0
PDSR	—	—	—	PDS4	PDS3	PDS2	PDS1	PDS0

Pin-shared Function Selection Register List

• IFS Register

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

Bit 7~1 Unimplemented, read as "0"

Bit 0 **INTPS**: INT input source pin selection
0: PA1
1: PD0

• PASR Register

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	—	—	PAS1	PAS0
R/W	—	—	—	—	—	—	R/W	R/W
POR	—	—	—	—	—	—	0	0

Bit 7~2 Unimplemented, read as "0"

Bit 1 **PAS1**: PA1 Pin-Shared function selection
0: PA1/INT
1: AN0

Bit 0 **PAS0**: PA0 Pin-Shared function selection
0: PA0
1: CTP1

• **PDSR Register**

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	PDS4	PDS3	PDS2	PDS1	PDS0
R/W	—	—	—	R/W	R/W	R/W	R/W	R/W
POR	—	—	—	0	0	0	0	0

Bit 7~5 Unimplemented, read as "0"

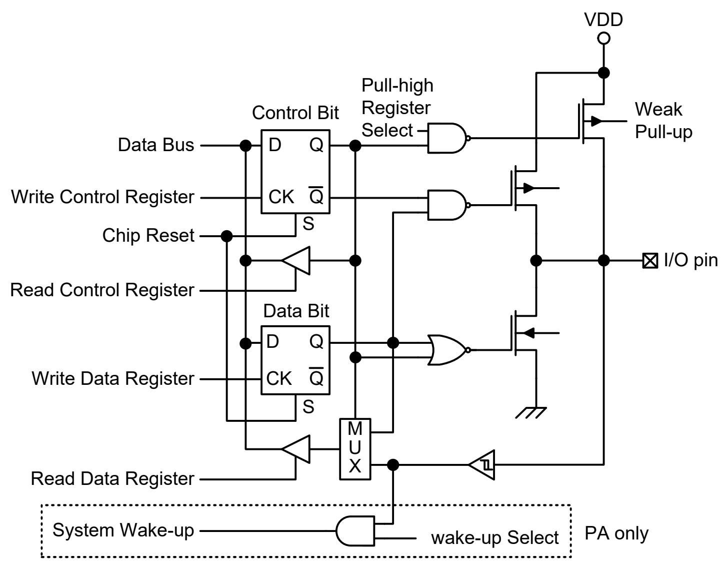
Bit 4~3 **PDS4~ PDS3**: PD2 Pin-Shared function selection
00: PD2
01: PD2
10: CTP0
11: AN3

Bit 2~1 **PDS2~ PDS1**: PD1 Pin-Shared function selection
00: PD1/STPI
01: PD1/STPI
10: STP
11: AN2

Bit 0 **PDS0**: PD0 Pin-Shared function selection
0: PD0/INT
1: AN1

I/O Pin Structures

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 functions. 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.

Timer Modules – TM

One of the most fundamental functions in any microcontroller device is the ability to control and measure time. To implement time related functions this device includes a Timer Module, abbreviated to the name TM. The TM is a multi-purpose timing unit and serves to provide operations such as Timer/Counter, Compare Match Output as well as being the functional unit for the generation of PWM signals. The TM has two individual interrupts. The addition of input and output pins for the TM ensures that users are provided with timing units with a wide and flexible range of features.

The common features of the different TM types are described here with more detailed information provided in the individual Compact and Standard TM sections.

Introduction

The device contains three TMs and each individual TM can be categorised as a certain type, namely Compact Type TM and Standard Type TM. Although similar in nature, the different TM types vary in their feature complexity. The common features to all of the Compact, Standard and Periodic TMs will be described in this section and the detailed operation regarding each of the TM types will be described in separate sections. The main features and differences between the two types of TMs are summarised in the accompanying table.

TM Function	CTM	STM
Timer/Counter	√	√
Input Capture	—	√
Compare Match Output	√	√
PWM Output	√	√
Single Pulse Output	—	√
PWM Alignment	Edge	Edge
PWM Adjustment Period & Duty	Duty or Period	Duty or Period

TM Function Summary

TM Operation

The different types of TM offer a diverse range of functions, from simple timing operations to PWM signal generation. The key to understanding how the TM operates is to see it in terms of a free running count-up counter whose value is then compared with the value of pre-programmed internal comparators. When the free running count-up counter has the same value as the pre-programmed comparator, known as a compare match situation, a TM interrupt signal will be generated which can clear the counter and perhaps also change the condition of the TM output pin. The internal TM counter is driven by a user selectable clock source, which can be an internal clock or an external pin.

TM Clock Source

The clock source which drives the main counter in each TM can originate from various sources. The selection of the required clock source is implemented using the CTnCK2~CTnCK0 bits in the CTMnC0 (n=0~1) or STCK2~STCK0 bits in the STMC0 control register. The clock source can be a ratio of the system clock, f_{SYS}, or the internal high clock, f_H, the f_{SUB} clock source or the external xTCKn pin. The xTCKn pin clock source is used to allow an external signal to drive the xTMn as an external clock source for event counting.

TM Interrupts

The Compact Type and Standard Type TMs each have two internal interrupts, one for each of the internal comparator A or comparator P, which generate a TM interrupt when a compare match condition occurs. When a TM interrupt is generated it can be used to clear the counter and also to change the state of the TM output pin.

TM External Pins

Each of the TMs, irrespective of what type, has one TM input pin, with the label xTCKn. The xTMn input pin, xTCKn, is essentially a clock source for the xTMn and is selected using the xTnCK2~xTnCK0 bits in the xTMnC0 register. This external TM input pin allows an external clock source to drive the internal TM. The xTCKn input pin can be chosen to have either a rising or falling active edge. The STCK pin is also used as the external trigger input pin in single pulse output mode for the STM.

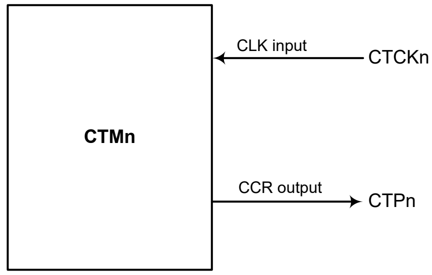
For the STM, another input pin, STPI, is the capture input whose active edge can be a rising edge, a falling edge or both rising and falling edges and the active edge transition type is selected using the STIO1~STIO0 bits in the STMC1 register.

The TMs each have one output pin, xTPn. When the TM is in the Compare Match Output Mode, these pins can be controlled by the TM to switch to a high or low level or to toggle when a compare match situation occurs. The external xTPn output pin is also the pin where the TM generates the PWM output waveform.

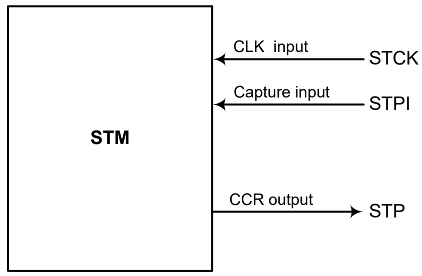
As the TM input and output pins are pin-shared with other functions, the TM input and output functions must first be setup using the relevant pin-shared function selection bits described in the Pin-shared Function section. The details of the pin-shared function selection are described in the pin-shared function section.

CTM0		CTM1		STM	
Input	Output	Input	Output	Input	Output
CTCK0	CTP0	CTCK1	CTP1	STCK, STPI	STP

TM External Pins



CTMn Function Pin Block Diagram (n=0~1)

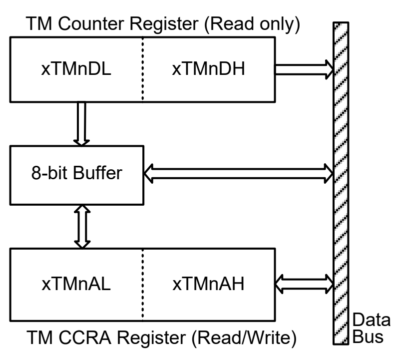


STM Function Pin Block Diagram

Programming Considerations

The TM Counter registers and the Capture/Compare CCRA registers, all have a low and high byte structure. The high bytes can be directly accessed, but as the low bytes can only be accessed via an internal 8-bit buffer, reading or writing to these register pairs must be carried out in a specific way. The important point to note is that data transfer to and from the 8-bit buffer and its related low byte only takes place when a write or read operation to its corresponding high byte is executed.

As the CCRA registers are implemented in the way shown in the following diagram and accessing these register pairs is carried out in a specific way as described above, it is recommended to use the "MOV" instruction to access the CCRA low byte registers, named xTMnAL, using the following access procedures. Accessing the CCRA low byte registers without following these access procedures will result in unpredictable values.



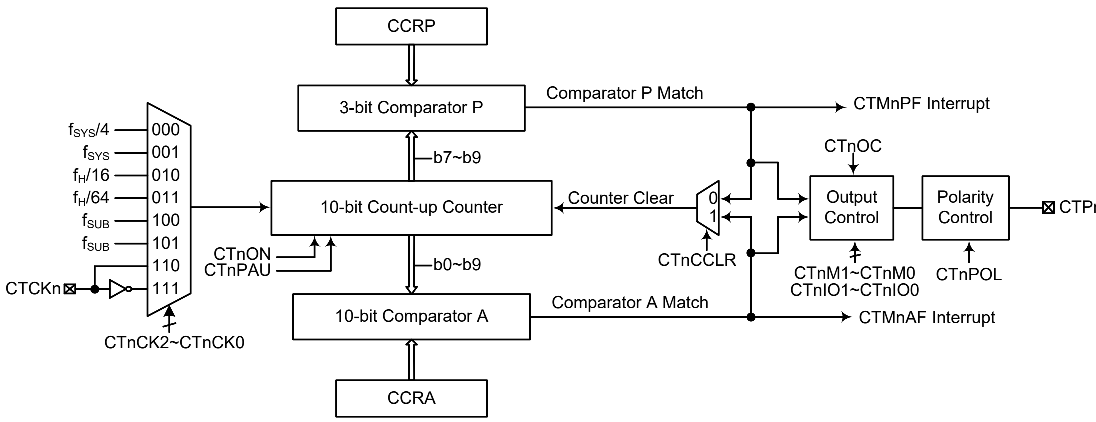
The following steps show the read and write procedures:

- Writing Data to CCRA
  - Step 1. Write data to Low Byte xTMnAL
    – note that here data is only written to the 8-bit buffer.
  - Step 2. Write data to High Byte xTMnAH
    – here data is written directly to the high byte registers and simultaneously data is latched from the 8-bit buffer to the Low Byte registers.

- Reading Data from the Counter Registers and CCRA
  - Step 1. Read data from the High Byte xTMnDH or xTMnAH
    – here data is read directly from the High Byte registers and simultaneously data is latched from the Low Byte register into the 8-bit buffer.
  - Step 2. Read data from the Low Byte xTMnDL or xTMnAL
    – this step reads data from the 8-bit buffer.

Compact Type TM – CTM

Although the simplest form of the three TM types, the Compact type TM still contains three operating modes, which are Compare Match Output, Timer/Event Counter and PWM Output modes. The Compact type TM can also be controlled with an external input pin and can drive one external output pin.



Note: The CTMn external pins may be pin-shared with other functions, so before using the CTMn, ensure that the pin-shared function registers have been set properly to enable the CTMn pin function.

Compact Type TM Block Diagram (n=0~1)

Compact Type TM Operation

At its core is a 10-bit count-up counter which is driven by a user selectable internal or external clock source. There are also two internal comparators with the names, Comparator A and Comparator P. These comparators will compare the value in the counter with CCRP and CCRA registers. The CCRP is three bits wide whose value is compared with the highest three bits in the counter while the CCRA is the ten bits and therefore compares with all counter bits.

The only way of changing the value of the 10-bit counter using the application program, is to clear the counter by changing the CTnON bit from low to high. The counter will also be cleared automatically by a counter overflow or a compare match with one of its associated comparators. When these conditions occur, a CTMn interrupt signal will also usually be generated. The Compact Type TM can operate in a number of different operational modes, can be driven by different clock sources including an input pin and can also control two output pins. All operating setup conditions are selected using relevant internal registers.

Compact Type TM Register Description

Overall operation of the Compact type TM is controlled using several registers. A read only register pair exists to store the internal counter 10-bit value, while a read/write register pair exists to store the internal 10-bit CCRA value. The remaining two registers are control registers which setup the different operating and control modes as well as the three CCRP bits.

Register Name	Bit							
	7	6	5	4	3	2	1	0
CTMnC0	CTnPAU	CTnCK2	CTnCK1	CTnCK0	CTnON	CTnRP2	CTnRP1	CTnRP0
CTMnC1	CTnM1	CTnM0	CTnIO1	CTnIO0	CTnOC	CTnPOL	CTnDPX	CTnCCLR
CTMnDL	D7	D6	D5	D4	D3	D2	D1	D0
CTMnDH	—	—	—	—	—	—	D9	D8
CTMnAL	D7	D6	D5	D4	D3	D2	D1	D0
CTMnAH	—	—	—	—	—	—	D9	D8

10-bit Compact Type TM Register List (n=0~1)

• **CTMnC0 Register**

Bit	7	6	5	4	3	2	1	0
Name	CTnPAU	CTnCK2	CTnCK1	CTnCK0	CTnON	CTnRP2	CTnRP1	CTnRP0
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 **CTnPAU**: CTMn Counter Pause Control
0: Run
1: Pause

The counter can be paused by setting this bit high. Clearing the bit to zero restores normal counter operation. When in a Pause condition the CTMn will remain powered up and continue to consume power. The counter will retain its residual value when this bit changes from low to high and resume counting from this value when the bit changes to a low value again.

Bit 6~4 **CTnCK2~CTnCK0**: Select CTMn Counter clock
000: $f_{SYS}/4$
001: f_{SYS}
010: $f_H/16$
011: $f_H/64$
100: f_{SUB}
101: f_{SUB}
110: CTCKn rising edge clock
111: CTCKn falling edge clock

These three bits are used to select the clock source for the CTMn. The external pin clock source can be chosen to be active on the rising or falling edge. The clock source f_{SYS} is the system clock, while f_H and f_{SUB} are other internal clocks, the details of which can be found in the “Operating Modes and System Clocks” section.

Bit 3 **CTnON**: CTMn Counter On/Off Control
0: Off
1: On

This bit controls the overall on/off function of the CTMn. Setting the bit high enables the counter to run, clearing the bit to 0 disables the CTMn. Clearing this bit to zero will stop the counter from counting and turn off the CTMn which will reduce its power consumption. When the bit changes state from low to high the internal counter value will be reset to zero, however when the bit changes from high to low, the internal counter will retain its residual value until the bit returns high again.

If the CTMn is in the Compare Match Output Mode or the PWM Output Mode then the CTMn output pin will be reset to its initial condition, as specified by the CTnOC bit, when the CTnON bit changes from low to high.

Bit 2~0 **CTnRP2~CTnRP0**: CTMn CCRP 3-bit register, compared with the CTMn Counter bit 9~bit 7

Comparator P Match Period
000: 1024 CTMn clocks
001: 128 CTMn clocks
010: 256 CTMn clocks
011: 384 CTMn clocks
100: 512 CTMn clocks
101: 640 CTMn clocks
110: 768 CTMn clocks
111: 896 CTMn clocks

These three bits are used to setup the value on the internal CCRP 3-bit register, which are then compared with the internal counter's highest three bits. The result of this comparison can be selected to clear the internal counter if the CTnCCLR bit is set to zero. Setting the CTnCCLR bit to zero ensures that a compare match with the CCRP values will reset the internal counter. As the CCRP bits are only compared with the highest three counter bits, the compare values exist in 128 clock cycle multiples. Clearing all three bits to zero is in effect allowing the counter to overflow at its maximum value.

• CTMnC1 Register

Bit	7	6	5	4	3	2	1	0
Name	CTnM1	CTnM0	CTnIO1	CTnIO0	CTnOC	CTnPOL	CTnDPX	CTnCCLR
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 **CTnM1~CTnM0**: Select CTMn Operating Mode
00: Compare Match Output Mode
01: Undefined
10: PWM Output Mode
11: Timer/Counter Mode

These bits setup the required operating mode for the CTMn. To ensure reliable operation the CTMn should be switched off before any changes are made to the CTnM1 and CTnM0 bits. In the Timer/Counter Mode, the CTMn output pin state is undefined.

Bit 5~4 **CTnIO1~CTnIO0**: Select CTPn output function

Compare Match Output Mode
00: No change
01: Output low
10: Output high
11: Toggle output

PWM Output Mode
00: PWM Output inactive state
01: PWM Output active state
10: PWM output
11: Undefined

Timer/Counter Mode
Unused

These two bits are used to determine how the CTMn output pin changes state when a certain condition is reached. The function that these bits select depends upon in which mode the CTMn is running.

In the Compare Match Output Mode, the CTnIO1 and CTnIO0 bits determine how the CTMn output pin changes state when a compare match occurs from the Comparator A. The CTMn output pin can be setup to switch high, switch low or to toggle its present state when a compare match occurs from the Comparator A. When the bits are both zero, then no change will take place on the output. The initial value of the CTMn output pin should be setup using the CTnOC bit in the CTMnC1 register. Note that the output level requested by the CTnIO1 and CTnIO0 bits must be different from the initial value setup using the CTnOC bit otherwise no change will occur on the CTMn output pin when a compare match occurs. After the CTMn output pin changes state it can be reset to its initial level by changing the level of the CTnON bit from low to high.

In the PWM Output Mode, the CTnIO1 and CTnIO0 bits determine how the CTMn output pin changes state when a certain compare match condition occurs. The PWM output function is modified by changing these two bits. It is necessary to only change the values of the CTnIO1 and CTnIO0 bits only after the CTMn has been switched off. Unpredictable PWM outputs will occur if the CTnIO1 and CTnIO0 bits are changed when The CTMn is running.

Bit 3 **CTnOC**: CTPn Output control bit

Compare Match Output Mode
0: Initial low
1: Initial high

PWM Output Mode
0: Active low
1: Active high

This is the output control bit for the CTMn output pin. Its operation depends upon whether CTMn is being used in the Compare Match Output Mode or in the PWM Output Mode. It has no effect if the CTMn is in the Timer/Counter Mode. In the Compare Match Output Mode it determines the logic level of the CTMn output pin before a compare match occurs. In the PWM Output Mode it determines if the PWM signal is active high or active low.

Bit 2 **CTnPOL**: CTPn Output polarity Control
0: Non-invert
1: Invert

This bit controls the polarity of the CTPn output pin. When the bit is set high the CTMn output pin will be inverted and not inverted when the bit is zero. It has no effect if the CTMn is in the Timer/Counter Mode.

Bit 1 **CTnDPX**: CTMn PWM period/duty Control
0: CCRP - period; CCRA - duty
1: CCRP - duty; CCRA - period

This bit determines which of the CCRA and CCRP registers are used for period and duty control of the PWM waveform.

Bit 0 **CTnCCLR**: Select CTMn Counter clear condition
0: CTMn Comparatror P match
1: CTMn Comparatror A match

This bit is used to select the method which clears the counter. Remember that the Compact TM contains two comparators, Comparator A and Comparator P, either of which can be selected to clear the internal counter. With the CTnCCLR bit set high, the counter will be cleared when a compare match occurs from the Comparator A. When the bit is low, the counter will be cleared when a compare match occurs from the Comparator P or with a counter overflow. A counter overflow clearing method can only be implemented if the CCRP bits are all cleared to zero. The CTnCCLR bit is not used in the PWM Output Mode.

• CTMnDL Register

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	0	0	0	0	0	0	0	0

Bit 7~0 CTMn Counter Low Byte Register bit 7 ~ bit 0
CTMn 10-bit Counter bit 7 ~ bit 0

• **CTMnDH Register**

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	—	—	D9	D8
R/W	—	—	—	—	—	—	R	R
POR	—	—	—	—	—	—	0	0

Bit 7~2 Unimplemented, read as "0"

Bit 1~0 CTMn Counter High Byte Register bit 1 ~ bit 0
CTMn 10-bit Counter bit 9 ~ bit 8

• **CTMnAL 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 CTMn CCRA Low Byte Register bit 7 ~ bit 0
CTMn 10-bit CCRA bit 7 ~ bit 0

• **CTMnAH Register**

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	—	—	D9	D8
R/W	—	—	—	—	—	—	R/W	R/W
POR	—	—	—	—	—	—	0	0

Bit 7~2 Unimplemented, read as "0"

Bit 1~0 CTMn CCRA High Byte Register bit 1 ~ bit 0
CTMn 10-bit CCRA bit 9 ~ bit 8

Compact Type TM Operating Modes

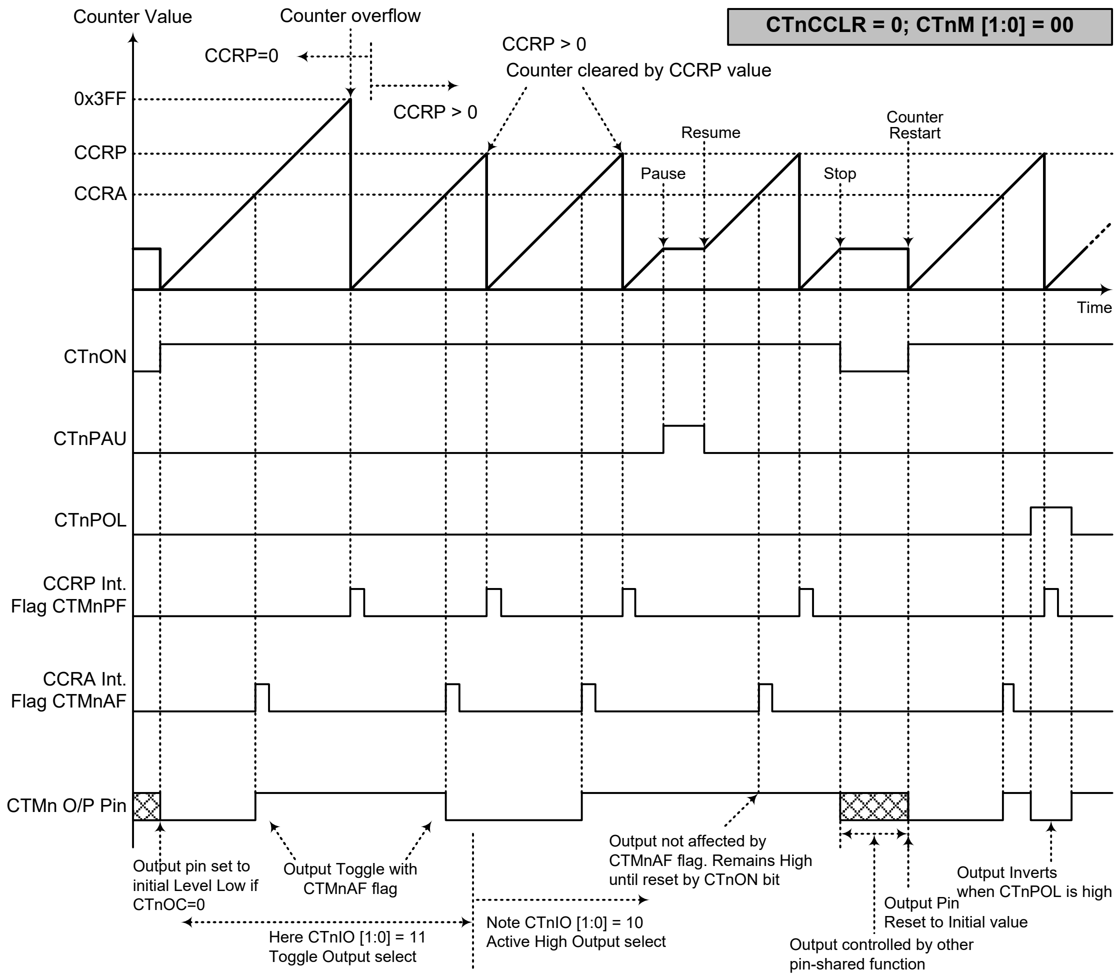
The Compact Type TM can operate in one of three operating modes, Compare Match Output Mode, PWM Output Mode or Timer/Counter Mode. The operating mode is selected using the CTnM1 and CTnM0 bits in the CTMnC1 register.

Compare Match Output Mode

To select this mode, bits CTnM1 and CTnM0 in the CTMnC1 register, should be set to 00 respectively. In this mode once the counter is enabled and running it can be cleared by three methods. These are a counter overflow, a compare match from Comparator A and a compare match from Comparator P. When the CTnCCLR bit is low, there are two ways in which the counter can be cleared. One is when a compare match from Comparator P, the other is when the CCRP bits are all zero which allows the counter to overflow. Here both CTMnAF and CTMnPF interrupt request flags for Comparator A and Comparator P respectively, will both be generated.

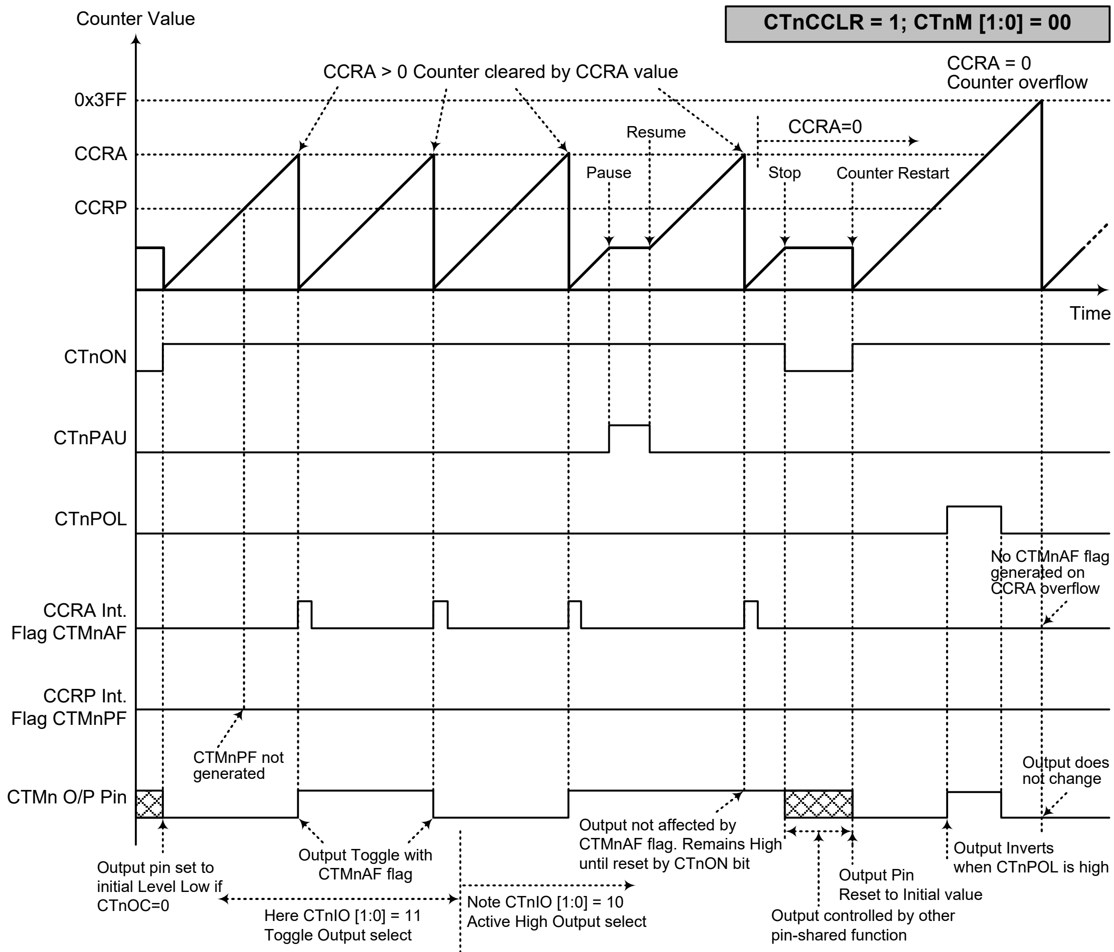
If the CTnCCLR bit in the CTMnC1 register is high then the counter will be cleared when a compare match occurs from Comparator A. However, here only the CTMnAF interrupt request flag will be generated even if the value of the CCRP bits is less than that of the CCRA registers. Therefore when CTnCCLR is high no CTMnPF interrupt request flag will be generated. If the CCRA bits are all zero, the counter will overflow when it reaches its maximum 10-bit, 3FF Hex, value, however here the CTMnAF interrupt request flag will not be generated.

As the name of the mode suggests, after a comparison is made, the CTMn output pin will change state. The CTMn output pin condition however only changes state when a CTMnAF interrupt request flag is generated after a compare match occurs from Comparator A. The CTMnPF interrupt request flag, generated from a compare match occurs from Comparator P, will have no effect on the CTMn output pin. The way in which the CTMn output pin changes state are determined by the condition of the CTnIO1 and CTnIO0 bits in the CTMnC1 register. The CTMn output pin can be selected using the CTnIO1 and CTnIO0 bits to go high, to go low or to toggle from its present condition when a compare match occurs from Comparator A. The initial condition of the CTMn output pin, which is setup after the CTnON bit changes from low to high, is setup using the CTnOC bit. Note that if the CTnIO1 and CTnIO0 bits are zero then no pin change will take place.



Compare Match Output Mode – CTnCCLR=0 (n=0~1)

Note: 1. With CTnCCLR=0, a Comparator P match will clear the counter
2. The CTMn output pin controlled only by the CTMnAF flag
3. The output pin reset to initial state by a CTnON bit rising edge



Compare Match Output Mode – CTnCCLR=1 (n=0~1)

Note: 1. With CTnCCLR=1, a Comparator A match will clear the counter
2. The CTMn output pin controlled only by the CTMnAF flag
3. The output pin reset to initial state by a CTnON rising edge
4. The CTMnPF flags is not generated when CTnCCLR=1

Timer/Counter Mode

To select this mode, bits CTnM1 and CTnM0 in the CTMnC1 register should be set to 11 respectively. The Timer/Counter Mode operates in an identical way to the Compare Match Output Mode generating the same interrupt flags. The exception is that in the Timer/Counter Mode the CTMn output pin is not used. Therefore the above description and Timing Diagrams for the Compare Match Output Mode can be used to understand its function. As the CTMn output pin is not used in this mode, the pin can be used as a normal I/O pin or other pin-shared function.

PWM Output Mode

To select this mode, bits CTnM1 and CTnM0 in the CTMnC1 register should be set to 10 respectively. The PWM function within the CTMn is useful for applications which require functions such as motor control, heating control, illumination control etc. By providing a signal of fixed frequency but of varying duty cycle on the CTMn output pin, a square wave AC waveform can be generated with varying equivalent DC RMS values.

As both the period and duty cycle of the PWM waveform can be controlled, the choice of generated waveform is extremely flexible. In the PWM Output Mode, the CTnCCLR bit has no effect on the PWM operation. Both of the CCRA and CCRP registers are used to generate the PWM waveform, one register is used to clear the internal counter and thus control the PWM waveform frequency, while the other one is used to control the duty cycle. Which register is used to control either frequency or duty cycle is determined using the CTnDPX bit in the CTMnC1 register. The PWM waveform frequency and duty cycle can therefore be controlled by the values in the CCRA and CCRP registers.

An interrupt flag, one for each of the CCRA and CCRP, will be generated when a compare match occurs from either Comparator A or Comparator P. The CTnOC bit In the CTMnC1 register is used to select the required polarity of the PWM waveform while the two CTnIO1 and CTnIO0 bits are used to enable the PWM output or to force the CTMn output pin to a fixed high or low level. The CTnPOL bit is used to reverse the polarity of the PWM output waveform.

- **10-bit CTMn, PWM Output Mode, Edge-aligned Mode, CTnDPX=0**

CCRP	1~7	0
Period	CCRP×128	1024
Duty	CCRA	

If f_{SYS}=8MHz, CTMn clock source is $f_{SYS}/4$, CCRP=2, CCRA=128.

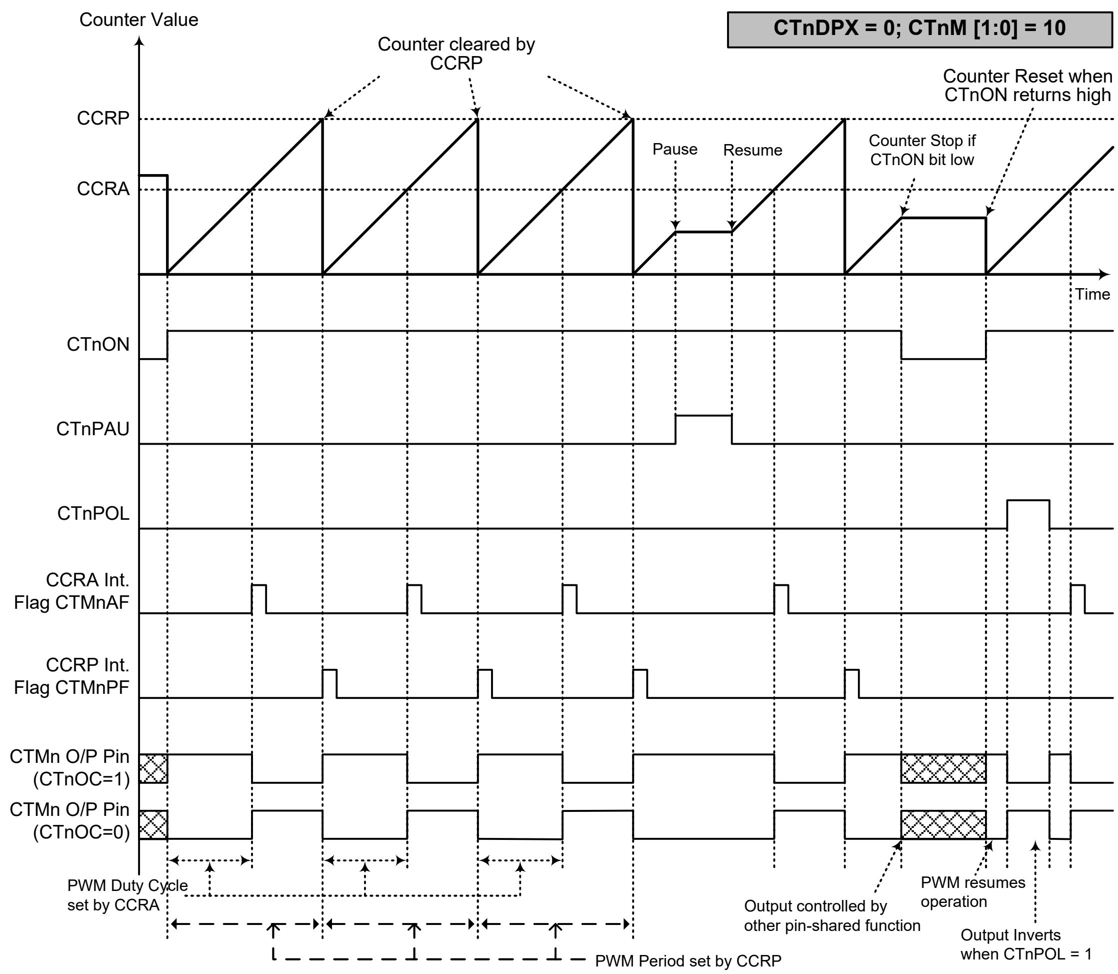
The CTMn PWM output frequency=$(f_{SYS}/4)/(2\times128)=f_{SYS}/1024$=3.9kHz, duty=128/(2×128)=50%.

If the Duty value defined by the CCRA register is equal to or greater than the Period value, then the PWM output duty is 100%.

- **10-bit CTMn, PWM Output Mode, Edge-aligned Mode, CTnDPX=1**

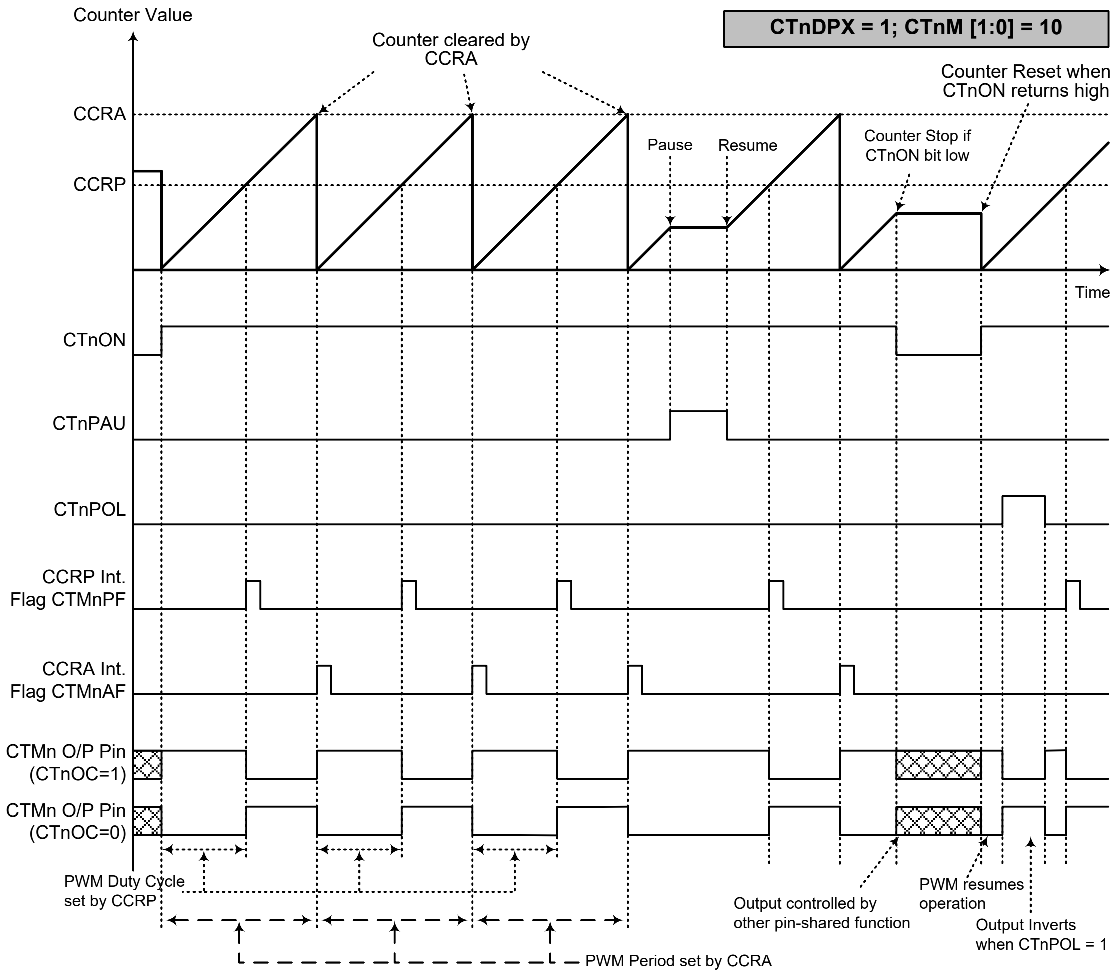
CCRP	1~7	0
Period	CCRA	
Duty	CCRP×128	1024

The PWM output period is determined by the CCRA register value together with the CTMn clock while the PWM duty cycle is defined by the CCRP register value.



PWM Output Mode – CTnDPX=0 (n=0~1)

Note: 1. Here CTnDPX=0 – Counter cleared by CCRP
2. A counter clear sets PWM Period
3. The internal PWM function continues running even when CTnIO[1:0]=00 or 01
4. The CTnCCLR bit has no influence on PWM operation

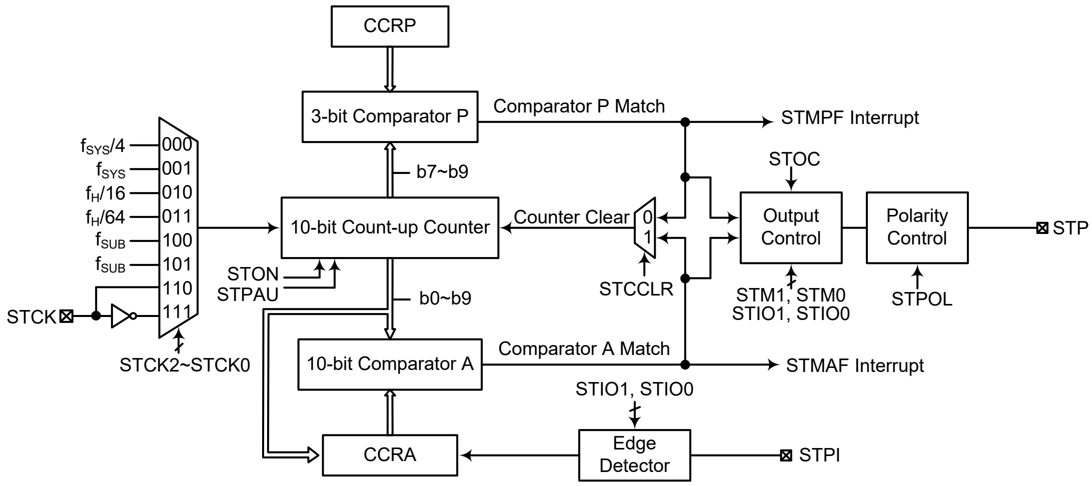


PWM Output Mode – CTnDPX=1 (n=0~1)

Note: 1. Here CTnDPX=1 – Counter cleared by CCRA
2. A counter clear sets PWM Period
3. The internal PWM function continues even when CTnIO[1:0]=00 or 01
4. The CTnCCLR bit has no influence on PWM operation

Standard Type TM – STM

The Standard Type TM contains five operating modes, which are Compare Match Output, Timer/Event Counter, Capture Input, Single Pulse Output and PWM Output modes. The Standard TM can also be controlled with two external input pins and can drive one external output pin.



Note: The STM external pins may be pin-shared with other functions, so before using the STM, ensure that the pin-shared function registers have been set properly to enable the STM pin function.

Standard Type TM Block Diagram

Standard TM Operation

The Standard TM is 10-bit wide and its core is a 10-bit count-up counter which is driven by a user selectable internal or external clock source. There are also two internal comparators with the names, Comparator A and Comparator P. These comparators will compare the value in the counter with CCRP and CCRA registers. The CCRP comparator is 3-bit wide whose value is compared with the highest 3 bits in the counter while the CCRA is the ten bits and therefore compares all counter bits.

The only way of changing the value of the 10-bit counter using the application program, is to clear the counter by changing the STON bit from low to high. The counter will also be cleared automatically by a counter overflow or a compare match with one of its associated comparators. When these conditions occur, a STM interrupt signal will also usually be generated. The Standard Type TM can operate in a number of different operational modes, can be driven by different clock sources including an input pin and can also control the output pin. All operating setup conditions are selected using relevant internal registers.

Standard Type TM Register Description

Overall operation of the Standard TM is controlled using a series of registers. A read only register pair exists to store the internal counter 10-bit value, while a read/write register pair exists to store the internal 10-bit CCRA value. The remaining two registers are control registers which setup the different operating and control modes as well as the 3-bit CCRP value.

Register Name	Bit							
	7	6	5	4	3	2	1	0
STMC0	STPAU	STCK2	STCK1	STCK0	STON	STRP2	STRP1	STRP0
STMC1	STM1	STM0	STIO1	STIO0	STOC	STPOL	STDPX	STCCLR
STMDL	D7	D6	D5	D4	D3	D2	D1	D0
STMDH	—	—	—	—	—	—	D9	D8
STMAL	D7	D6	D5	D4	D3	D2	D1	D0
STMAH	—	—	—	—	—	—	D9	D8

10-bit Standard TM Register List

• STMC0 Register

Bit	7	6	5	4	3	2	1	0
Name	STPAU	STCK2	STCK1	STCK0	STON	STRP2	STRP1	STRP0
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 **STPAU**: STM Counter Pause control
0: Run
1: Pause

The counter can be paused by setting this bit high. Clearing the bit to zero restores normal counter operation. When in a Pause condition the STM will remain powered up and continue to consume power. The counter will retain its residual value when this bit changes from low to high and resume counting from this value when the bit changes to a low value again.

Bit 6~4 **STCK2~STCK0**: Select STM Counter clock
000: $f_{SYS}/4$
001: f_{SYS}
010: $f_H/16$
011: $f_H/64$
100: f_{SUB}
101: f_{SUB}
110: STCK rising edge clock
111: STCK falling edge clock

These three bits are used to select the clock source for the STM. The external pin clock source can be chosen to be active on the rising or falling edge. The clock source f_{SYS} is the system clock, while f_H and f_{SUB} are other internal clocks, the details of which can be found in the "Operating Modes and System Clocks" section.

Bit 3 **STON**: STM Counter On/Off control
0: Off
1: On

This bit controls the overall on/off function of the STM. Setting the bit high enables the counter to run while clearing the bit disables the STM. Clearing this bit to zero will stop the counter from counting and turn off the STM which will reduce its power consumption. When the bit changes state from low to high the internal counter value will be reset to zero, however when the bit changes from high to low, the internal counter will retain its residual value until the bit returns high again. If the STM is in the Compare Match Output Mode, PWM Output Mode or Single Pulse Output Mode then the STM output pin will be reset to its initial condition, as specified by the STOC bit, when the STON bit changes from low to high.

Bit 2~0 **STnRP2~STnRP0**: STM CCRP 3-bit register, compared with the STM Counter bit 9~bit 7

Comparator P Match Period
000: 1024 STM clocks
001: 128 STM clocks
010: 256 STM clocks
011: 384 STM clocks
100: 512 STM clocks
101: 640 STM clocks
110: 768 STM clocks
111: 896 STM clocks

These three bits are used to setup the value on the internal CCRP 3-bit register, which are then compared with the internal counter's highest three bits. The result of this comparison can be selected to clear the internal counter if the STCCLR bit is set to zero. Setting the STCCLR bit to zero ensures that a compare match with the CCRP values will reset the internal counter. As the CCRP bits are only compared with the highest three counter bits, the compare values exist in 128 clock cycle multiples. Clearing all three bits to zero is in effect allowing the counter to overflow at its maximum value.

- **STMC1 Register**

Bit	7	6	5	4	3	2	1	0
Name	STM1	STM0	STIO1	STIO0	STOC	STPOL	STDPX	STCCLR
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 **STM1~STM0**: Select STM Operating Mode
00: Compare Match Output Mode
01: Capture Input Mode
10: PWM Output Mode or Single Pulse Output Mode
11: Timer/Counter Mode

These bits setup the required operating mode for the STM. To ensure reliable operation the STM should be switched off before any changes are made to the STM1 and STM0 bits. In the Timer/Counter Mode, the STM output pin state is undefined.

Bit 5~4 **STIO1~STIO0**: Select STM external pin function

Compare Match Output Mode
00: No change
01: Output low
10: Output high
11: Toggle output

PWM Output Mode/Single Pulse Output Mode
00: PWM output inactive state
01: PWM output active state
10: PWM output
11: Single Pulse Output

Capture Input Mode
00: Input capture at rising edge of STPI
01: Input capture at falling edge of STPI
10: Input capture at rising/falling edge of STPI
11: Input capture disabled

Timer/Counter Mode
Unused

These two bits are used to determine how the STM external pin changes state when a certain condition is reached. The function that these bits select depends upon in which mode the STM is running.

In the Compare Match Output Mode, the STIO1 and STIO0 bits determine how the STM output pin changes state when a compare match occurs from the Comparator A. The STM output pin can be setup to switch high, switch low or to toggle its present state when a compare match occurs from the Comparator A. When the bits are both zero, then no change will take place on the output. The initial value of the STM output pin should be setup using the STOC bit in the STMC1 register. Note that the output level requested by the STIO1 and STIO0 bits must be different from the initial value setup using the STOC bit otherwise no change will occur on the STM output pin when a compare match occurs. After the STM output pin changes state, it can be reset to its initial level by changing the level of the STON bit from low to high.

In the PWM Output Mode, the STIO1 and STIO0 bits determine how the STM output pin changes state when a certain compare match condition occurs. The PWM output function is modified by changing these two bits. It is necessary to only change the values of the STIO1 and STIO0 bits only after the STM has been switched off. Unpredictable PWM outputs will occur if the STIO1 and STIO0 bits are changed when the STM is running.

Bit 3 **STOC**: STM STP Output control

Compare Match Output Mode
0: Initial low
1: Initial high

PWM Output Mode/Single Pulse Output Mode
0: Active low
1: Active high

This is the output control bit for the STM output pin. Its operation depends upon whether STM is being used in the Compare Match Output Mode or in the PWM Output Mode/Single Pulse Output Mode. It has no effect if the STM is in the Timer/Counter Mode. In the Compare Match Output Mode it determines the logic level of the STM output pin before a compare match occurs. In the PWM Output Mode it determines if the PWM signal is active high or active low. In the Single Pulse Output Mode it determines the logic level of the STM output pin when the STON bit changes from low to high.

Bit 2 **STPOL**: STM STP Output polarity control
0: Non-invert
1: Invert

This bit controls the polarity of the STP output pin. When the bit is set high the STM output pin will be inverted and not inverted when the bit is zero. It has no effect if the STM is in the Timer/Counter Mode.

Bit 1 **STDPX**: STM PWM duty/period control
0: CCRP – period; CCRA – duty
1: CCRP – duty; CCRA – period

This bit determines which of the CCRA and CCRP registers are used for period and duty control of the PWM waveform.

Bit 0 **STCCLR**: STM Counter Clear condition selection
0: Comparator P match
1: Comparator A match

This bit is used to select the method which clears the counter. Remember that the Standard TM contains two comparators, Comparator A and Comparator P, either of which can be selected to clear the internal counter. With the STCCLR bit set high, the counter will be cleared when a compare match occurs from the Comparator A. When the bit is low, the counter will be cleared when a compare match occurs from the Comparator P or with a counter overflow. A counter overflow clearing method can only be implemented if the CCRP bits are all cleared to zero. The STCCLR bit is not used in the PWM Output Mode, Single Pulse Output Mode or Capture Input Mode.

• **STMDL Register**

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	0	0	0	0	0	0	0	0

Bit 7~0 STM Counter Low Byte Register bit 7 ~ bit 0
STM 10-bit Counter bit 7 ~ bit 0

• **STMDH Register**

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	—	—	D9	D8
R/W	—	—	—	—	—	—	R	R
POR	—	—	—	—	—	—	0	0

Bit 7~2 Unimplemented, read as “0”

Bit 1~0 STM Counter High Byte Register bit 1 ~ bit 0
STM 10-bit Counter bit 9 ~ bit 8

• **STMAL 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 STM CCRA Low Byte Register bit 7 ~ bit 0
STM 10-bit CCRA bit 7 ~ bit 0

• **STMAH Register**

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	—	—	D9	D8
R/W	—	—	—	—	—	—	R/W	R/W
POR	—	—	—	—	—	—	0	0

Bit 7~2 Unimplemented, read as “0”

Bit 1~0 STM CCRA High Byte Register bit 1 ~ bit 0
STM 10-bit CCRA bit 9 ~ bit 8

Standard Type TM Operation Modes

The Standard Type TM can operate in one of five operating modes, Compare Match Output Mode, PWM Output Mode, Single Pulse Output Mode, Capture Input Mode or Timer/Counter Mode. The operating mode is selected using the STM1 and STM0 bits in the STMC1 register.

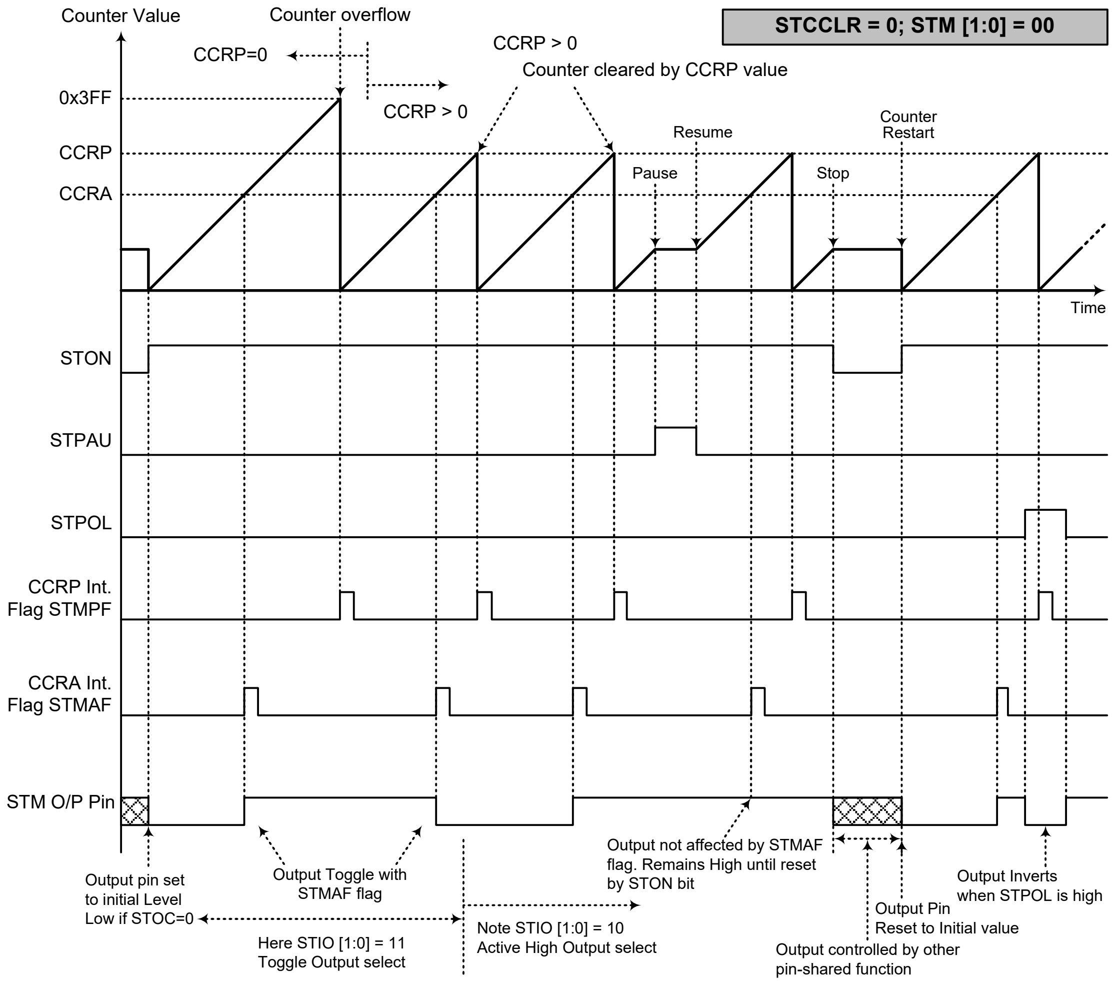
Compare Match Output Mode

To select this mode, bits STM1 and STM0 in the STMC1 register, should be set to 00 respectively. In this mode once the counter is enabled and running it can be cleared by three methods. These are a counter overflow, a compare match from Comparator A and a compare match from Comparator P. When the STCCLR bit is low, there are two ways in which the counter can be cleared. One is when a compare match from Comparator P, the other is when the CCRP bits are all zero which allows the counter to overflow. Here both STMAF and STMPF interrupt request flags for Comparator A and Comparator P respectively, will both be generated.

If the STCCLR bit in the STMC1 register is high then the counter will be cleared when a compare match occurs from Comparator A. However, here only the STMAF interrupt request flag will be generated even if the value of the CCRP bits is less than that of the CCRA registers. Therefore when STCCLR is high no STMPF interrupt request flag will be generated. In the Compare Match Output Mode, the CCRA cannot be set to “0”.

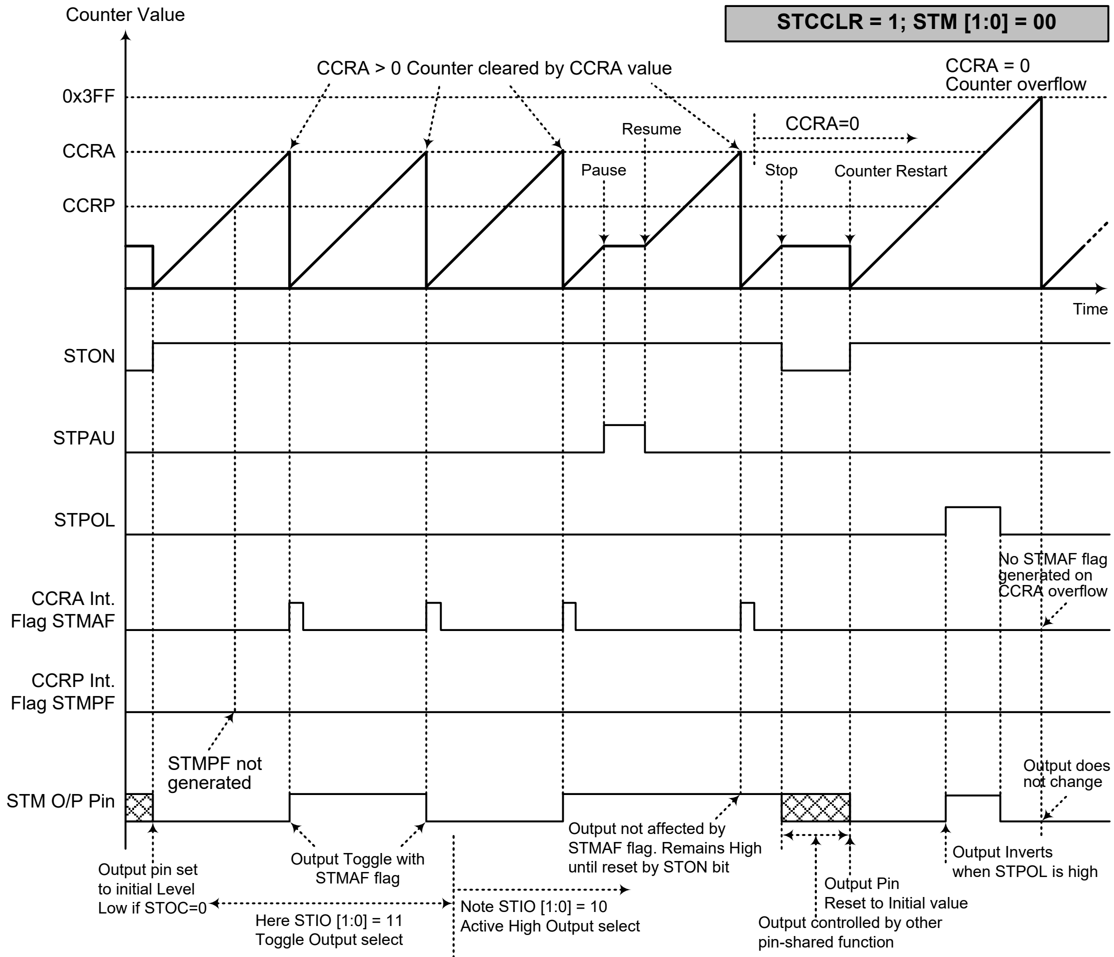
If the CCRA bits are all zero, the counter will overflow when it reaches its maximum 10-bit, 3FF Hex, value, however here the STMAF interrupt request flag will not be generated.

As the name of the mode suggests, after a comparison is made, the STM output pin, will change state. The STM output pin condition however only changes state when a STMAF interrupt request flag is generated after a compare match occurs from Comparator A. The STMPF interrupt request flag, generated from a compare match occurs from Comparator P, will have no effect on the STM output pin. The way in which the STM output pin changes state are determined by the condition of the STIO1 and STIO0 bits in the STMC1 register. The STM output pin can be selected using the STIO1 and STIO0 bits to go high, to go low or to toggle from its present condition when a compare match occurs from Comparator A. The initial condition of the STM output pin, which is setup after the STON bit changes from low to high, is setup using the STOC bit. Note that if the STIO1 and STIO0 bits are zero then no pin change will take place.



Compare Match Output Mode – STCCLR=0

Note: 1. With STCCLR=0 a Comparator P match will clear the counter
2. The STM output pin is controlled only by the STMAF flag
3. The output pin is reset to its initial state by a STON bit rising edge



Compare Match Output Mode – STCCLR=1

Note: 1. With STCCLR=1 a Comparator A match will clear the counter
2. The STM output pin is controlled only by the STMAF flag
3. The output pin is reset to its initial state by a STON bit rising edge
4. The STMPF flag is not generated when STCCLR=1

Timer/Counter Mode

To select this mode, bits STM1 and STM0 in the STMC1 register should be set to 11 respectively. The Timer/Counter Mode operates in an identical way to the Compare Match Output Mode generating the same interrupt flags. The exception is that in the Timer/Counter Mode the STM output pin is not used. Therefore the above description and Timing Diagrams for the Compare Match Output Mode can be used to understand its function. As the STM output pin is not used in this mode, the pin can be used as a normal I/O pin or other pin-shared functions.

PWM Output Mode

To select this mode, bits STM1 and STM0 in the STMC1 register should be set to 10 respectively and also the STIO1 and STIO0 bits should be set to 10 respectively. The PWM function within the STM is useful for applications which require functions such as motor control, heating control, illumination control etc. By providing a signal of fixed frequency but of varying duty cycle on the STM output pin, a square wave AC waveform can be generated with varying equivalent DC RMS values.

As both the period and duty cycle of the PWM waveform can be controlled, the choice of generated waveform is extremely flexible. In the PWM output mode, the STCCLR bit has no effect as the PWM period. Both of the CCRA and CCRP registers are used to generate the PWM waveform, one register is used to clear the internal counter and thus control the PWM waveform frequency, while the other one is used to control the duty cycle. Which register is used to control either frequency or duty cycle is determined using the STDPX bit in the STMC1 register. The PWM waveform frequency and duty cycle can therefore be controlled by the values in the CCRA and CCRP registers.

An interrupt flag, one for each of the CCRA and CCRP, will be generated when a compare match occurs from either Comparator A or Comparator P. The STOC bit in the STMC1 register is used to select the required polarity of the PWM waveform while the two STIO1 and STIO0 bits are used to enable the PWM output or to force the STM output pin to a fixed high or low level. The STPOL bit is used to reverse the polarity of the PWM output waveform.

- **10-bit STM, PWM Output Mode, Edge-aligned Mode, STDPX=0**

CCRP	1~7	0
Period	CCRP×128	1024
Duty	CCRA	

If f_{SYS}=8MHz, STM clock source is f_{SYS}/4, CCRP=4 and CCRA=128,

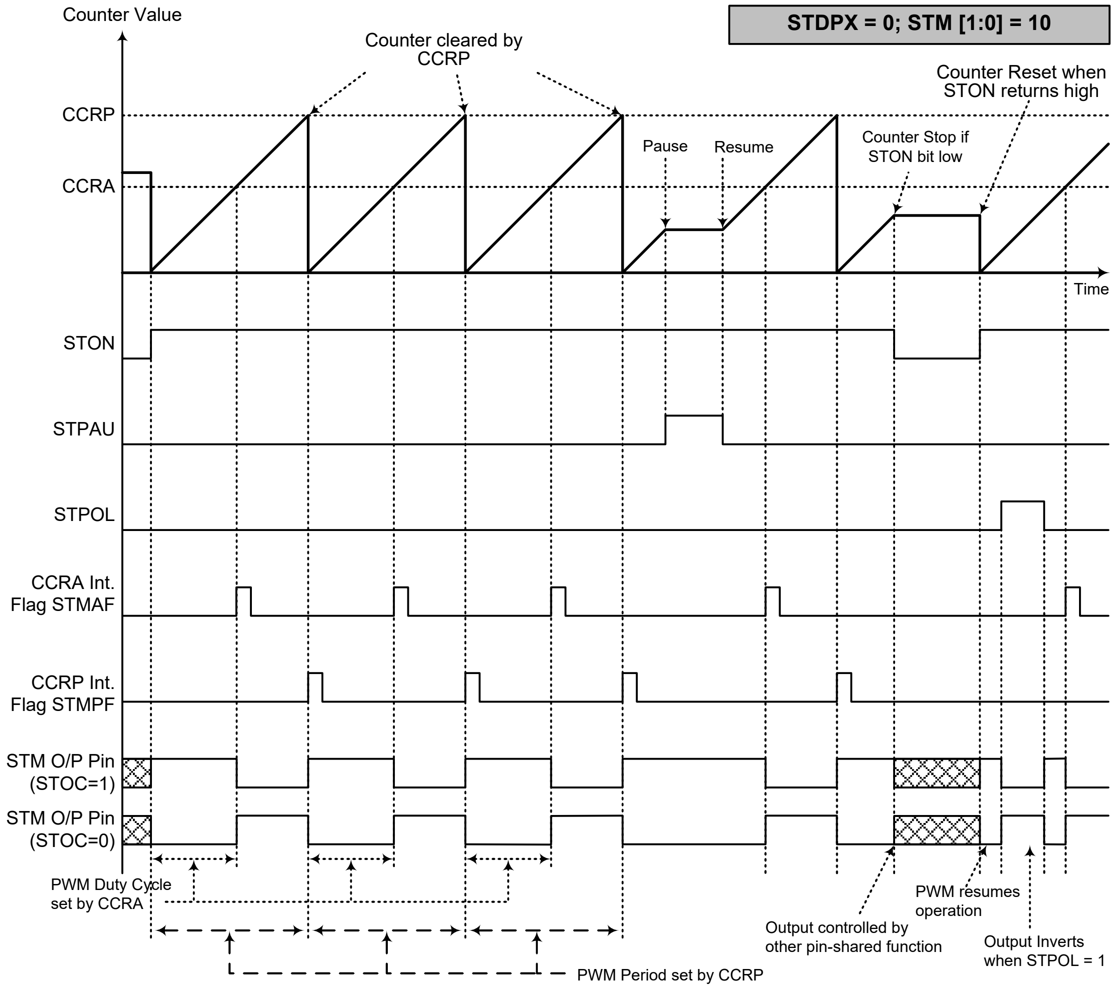
The STM PWM output frequency=(f_{SYS}/4)/(4×128)=f_{SYS}/2048=3.9kHz, duty=128/(4×128)=25%.

If the Duty value defined by the CCRA register is equal to or greater than the Period value, then the PWM output duty is 100%.

- **10-bit STM, PWM Output Mode, Edge-aligned Mode, STDPX=1**

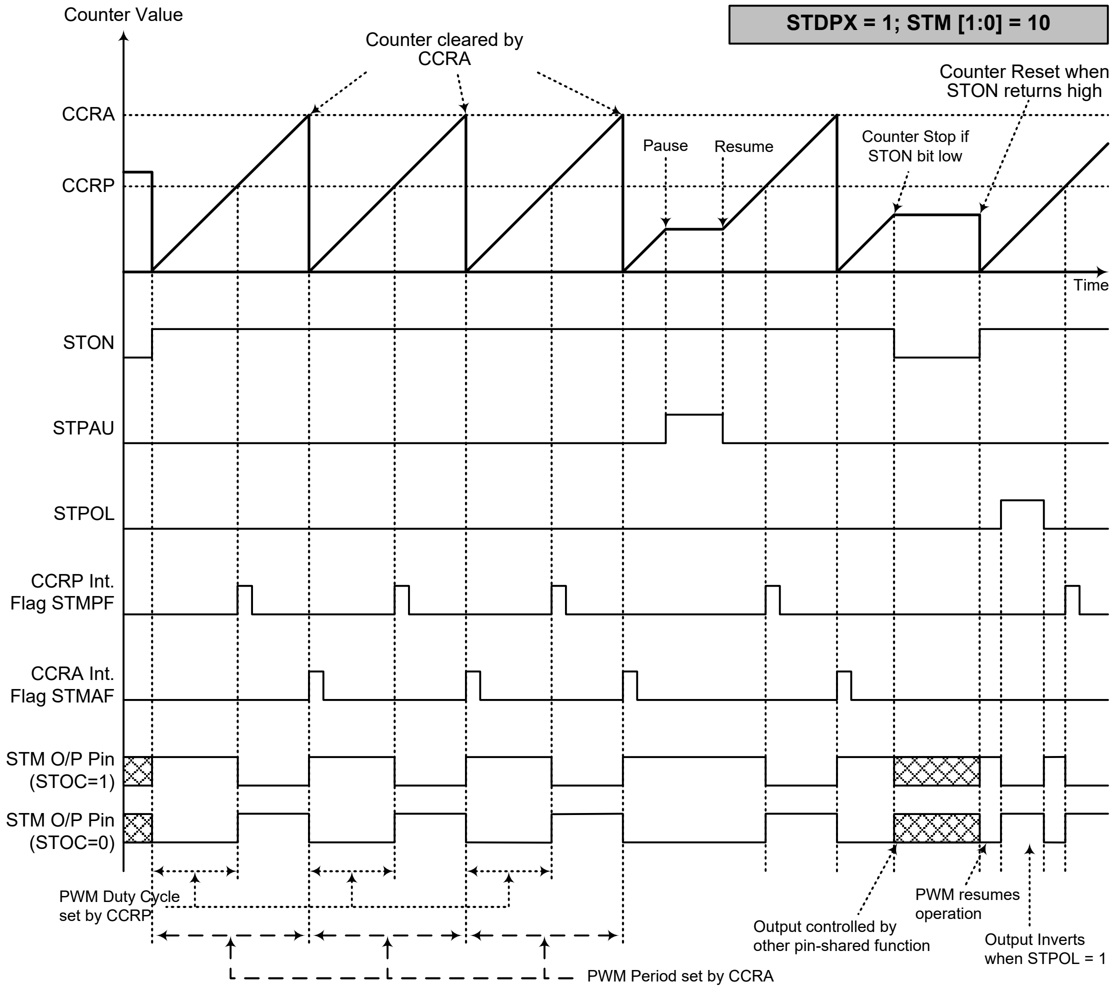
CCRP	1~7	0
Period	CCRA	
Duty	CCRP×128	1024

The PWM output period is determined by the CCRA register value together with the STM clock while the PWM duty cycle is defined by the CCRP register value except when the CCRP value is equal to 0.



PWM Output Mode – STDPX=0

Note: 1. Here STDPX=0 – Counter cleared by CCRP
2. A counter clear sets the PWM Period
3. The internal PWM function continues running even when STIO[1:0]=00 or 01
4. The STCCLR bit has no influence on PWM operation



PWM Output Mode – STDPX=1

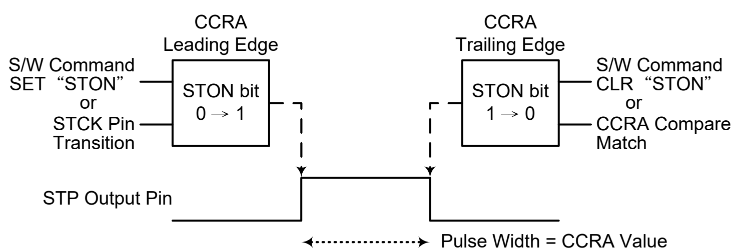
Note: 1. Here STDPX=1 – Counter cleared by CCRA
2. A counter clear sets the PWM Period
3. The internal PWM function continues even when STIO[1:0]=00 or 01
4. The STCCLR bit has no influence on PWM operation

Single Pulse Output Mode

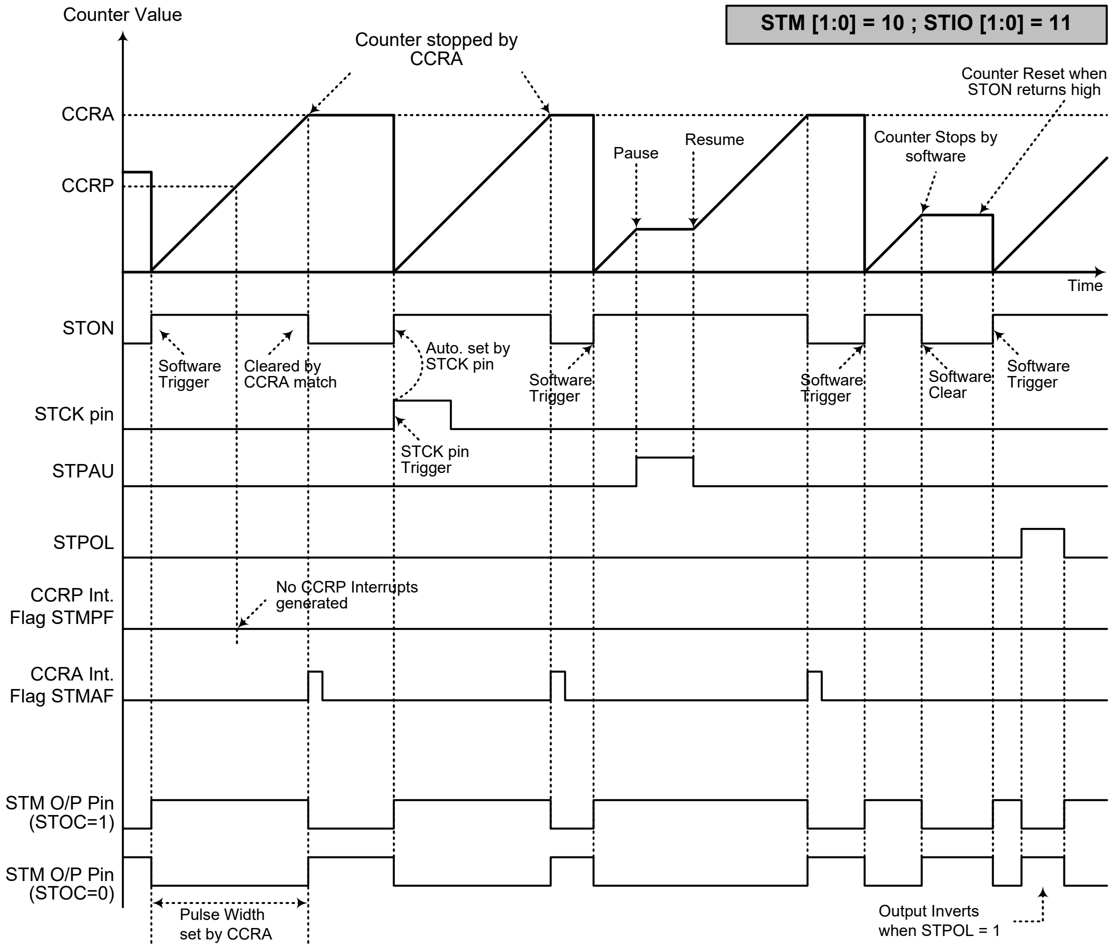
To select this mode, bits STM1 and STM0 in the STMC1 register should be set to 10 respectively and also the STIO1 and STIO0 bits should be set to 11 respectively. The Single Pulse Output Mode, as the name suggests, will generate a single shot pulse on the STM output pin.

The trigger for the pulse output leading edge is a low to high transition of the STON bit, which can be implemented using the application program. However in the Single Pulse Output Mode, the STON bit can also be made to automatically change from low to high using the external STCK pin, which will in turn initiate the Single Pulse output. When the STON bit transitions to a high level, the counter will start running and the pulse leading edge will be generated. The STON bit should remain high when the pulse is in its active state. The generated pulse trailing edge will be generated when the STON bit is cleared to zero, which can be implemented using the application program or when a compare match occurs from Comparator A.

However a compare match from Comparator A will also automatically clear the STON bit and thus generate the Single Pulse output trailing edge. In this way the CCRA value can be used to control the pulse width. A compare match from Comparator A will also generate a STM interrupt. The counter can only be reset back to zero when the STON bit changes from low to high when the counter restarts. In the Single Pulse Output Mode CCRP is not used. The STCCLR and STDPX bits are not used in this Mode.



Single Pulse Generation



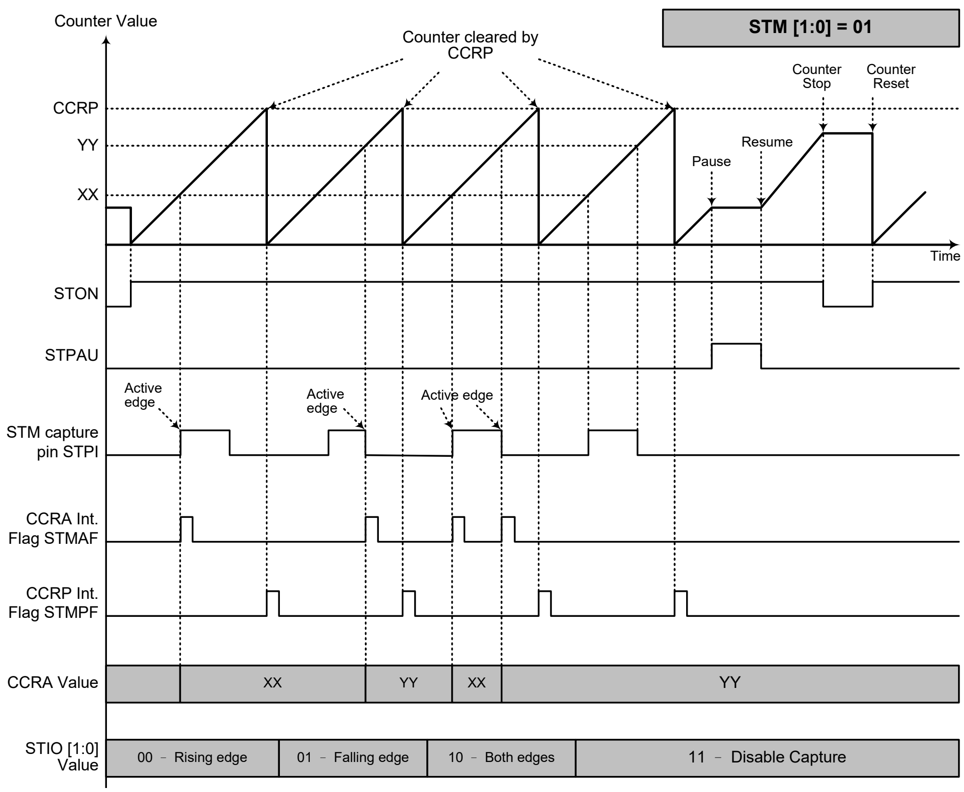
Single Pulse Output Mode

Note: 1. Counter stopped by CCRA
2. CCRP is not used
3. The pulse triggered by the STCK pin or by setting the STON bit high
4. A STCK pin active edge will automatically set the STON bit high
5. In the Single Pulse Output Mode, STIO[1:0] must be set to “11” and cannot be changed

Capture Input Mode

To select this mode bits STM1 and STM0 in the STMC1 register should be set to 01 respectively. This mode enables external signals to capture and store the present value of the internal counter and can therefore be used for applications such as pulse width measurements. The external signal is supplied on the STPI pin, whose active edge can be a rising edge, a falling edge or both rising and falling edges; the active edge transition type is selected using the STIO1 and STIO0 bits in the STMC1 register. The counter is started when the STON bit changes from low to high which is initiated using the application program.

When the required edge transition appears on the STPI pin the present value in the counter will be latched into the CCRA registers and a STM interrupt generated. Irrespective of what events occur on the STPI pin the counter will continue to free run until the STON bit changes from high to low. When a CCRP compare match occurs the counter will reset back to zero; in this way the CCRP value can be used to control the maximum counter value. When a CCRP compare match occurs from Comparator P, a STM interrupt will also be generated. Counting the number of overflow interrupt signals from the CCRP can be a useful method in measuring long pulse widths. The STIO1 and STIO0 bits can select the active trigger edge on the STPI pin to be a rising edge, falling edge or both edge types. If the STIO1 and STIO0 bits are both set high, then no capture operation will take place irrespective of what happens on the STPI pin, however it must be noted that the counter will continue to run. The STCCLR and STDPX bits are not used in this Mode.



Capture Input Mode

Note: 1. STM[1:0]=01 and active edge set by the STIO[1:0] bits
2. A STM Capture input pin active edge transfers the counter value to CCRA
3. STCCLR bit not used
4. No output function – STOC and STPOL bits are not used
5. CCRP determines the counter value and the counter has a maximum count value when CCRP is equal to zero

Analog to Digital Converter

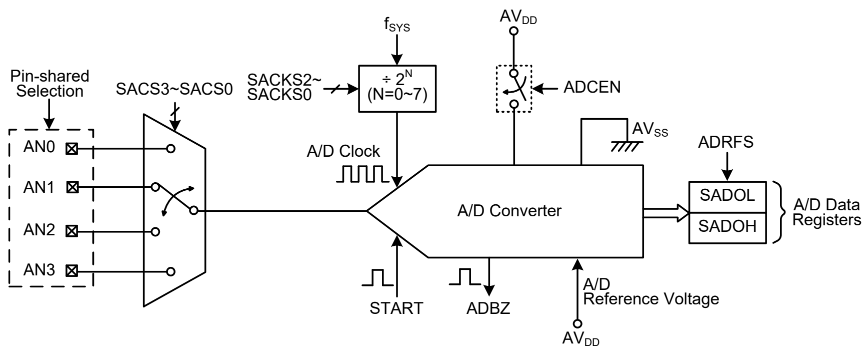
The need to interface to real world analog signals is a common requirement for many electronic systems. However, to properly process these signals by a microcontroller, they must first be converted into digital signals by A/D converters. By integrating the A/D conversion electronic circuitry into the 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 a 10-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 selection bit should first be properly configured and then the desired external channel input should be selected using the SACS3~SACS0 bits. More detailed information about the A/D input signal selection is described in the “A/D Converter Control Registers” and “A/D Converter Input Signals” sections respectively.

External Input Channels	A/D Channel Select Bits
AN0~AN3	SACS3~SACS0

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



A/D Converter Structure

A/D Converter Register Description

Overall operation of the A/D converter is controlled using four registers. A read only register pair exists to store the A/D converter data 10-bit value. The remaining two registers are control registers which setup the operating and control function of the A/D converter.

Register Name	Bit							
	7	6	5	4	3	2	1	0
SADOL (ADRFS=0)	D1	D0	—	—	—	—	—	—
SADOL (ADRFS=1)	D7	D6	D5	D4	D3	D2	D1	D0
SADOH (ADRFS=0)	D9	D8	D7	D6	D5	D4	D3	D2
SADOH (ADRFS=1)	—	—	—	—	—	—	D9	D8
SADC0	START	ADBZ	ADCEN	ADRFS	SACS3	SACS2	SACS1	SACS0
SADC1	—	—	—	—	—	SACKS2	SACKS1	SACKS0

A/D Converter Register List

A/D Converter Data Registers – SADOL, SADOH

As the internal A/D converter provides a 10-bit digital conversion value, it requires two data registers to store the converted value. These are a high byte register, known as SADOH, and a low byte register, known as SADOL. After the conversion process takes place, these registers can be directly read by the microcontroller to obtain the digitised conversion value. As only 10 bits of the 16-bit register space is utilised, the format in which the data is stored is controlled by the ADRFS bit in the SADC0 register, as shown in the accompanying table. D0~D9 are the conversion result data bits. Any unused bits will be read as zero. Note that A/D data registers contents will be unchanged if the A/D converter is disabled.

ADRFS	SADOH								SADOL							
	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
0	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0	0	0	0	0	0	0
1	0	0	0	0	0	0	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0

A/D Converter Data Registers

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 channel 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 analog signal inputs 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	ADRFS	SACS3	SACS2	SACS1	SACS0
R/W	R/W	R	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	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 1 to enable the A/D converter. If the bit is cleared to zero, 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 pair known as SADOH and SADOL will be unchanged.

Bit 4 **ADRFS**: A/D converter output data format selection
0: A/D converter data format → SADOH=D[9:2]; SADOL=D[1:0]
1: A/D converter data format → SADOH=D[9:8]; SADOL=D[7:0]
This bit controls the format of the 10-bit converted A/D value in the two A/D data registers. Details are provided in the A/D data register section.

Bit 3~0 **SACS3~SACS0**: A/D converter external analog channel input selection
0000: AN0
0001: AN1
0010: AN2
0011: AN3
0100~1111: undefined, input floating

- **SADC1 Register**

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

Bit 7~3 Unimplemented, read as “0”

Bit 2~0 **SACKS2~SACKS0**: A/D conversion clock 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 reference voltage supply to the A/D converter is from the internal A/D power supply voltage, AV_{DD}. The analog input values must not be allowed to exceed the value of this A/D reference voltage.

A/D Converter Input Signals

All the external A/D analog channel input pins are pin-shared with the I/O pins as well as other functions. The corresponding control bits for each A/D external input pin in the PASR and PDSR register determine whether the input pins are set as A/D converter analog inputs or whether they have other functions. If the pin is set to be as an A/D analog channel input, the original pin functions 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 pin-shared function control bits enable an A/D input, the status of the port control register will be overridden. The external input channel is selected using the SACS3~SACS0 bits in the SADC0 register.

A/D Converter Operation

The START bit in the SADC0 register is used to start the A/D 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 will be cleared to 0. In addition, the corresponding A/D interrupt request flag will be set in the interrupt control register, and if the associated interrupts are enabled, an appropriate internal interrupt signal will be generated. 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 larger 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.

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

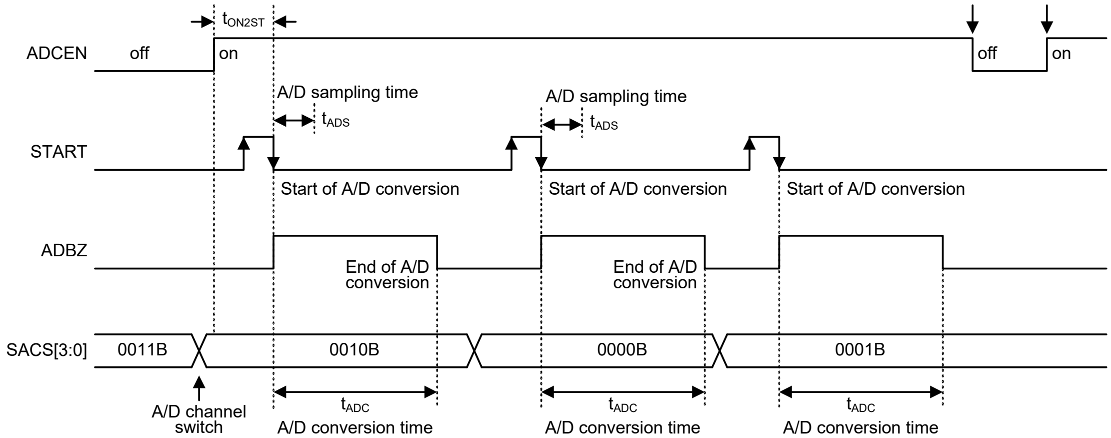
Controlling the power on/off function of the A/D converter circuitry is implemented using the ADCEN bit in the SADC0 register. This bit must be set high to power on the A/D converter. When the ADCEN bit is set high to power on the A/D converter internal circuitry a certain delay, as indicated in the timing diagram, must be allowed before an A/D conversion is initiated. Even if no pins are selected for use as A/D inputs, 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 which is defined as t_{ADS} takes 4 A/D clock cycles and the data conversion takes 10 A/D clock cycles. Therefore a total of 14 A/D clock cycles for an external input A/D conversion which is defined as t_{ADC} are necessary.

$$\text{Maximum single A/D conversion rate} = \text{A/D clock period} \div 14$$

The accompanying diagram shows graphically the various stages involved in an 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 the A/D conversion is 14 t_{ADCK} clock cycles where t_{ADCK} is equal to the A/D clock period.



A/D Conversion Timing

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 correctly programming bits SACKS2~SACKS0 in the SADC1 register.
- Step 2
  Enable the A/D converter by setting the ADCEN bit in the SADC0 register to “1”.
- Step 3
  Select which channel is to be connected to the internal A/D converter by correctly configuring the SACS bit field in the SADC0 register.
- Step 4
  Select A/D converter output data format by setting the ADRFS bit in the SADC0 register.
- Step 5
  If the 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 SADOH and SADOL registers.

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 0 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 10-bit A/D converter, its full-scale converted digitised value is equal to 3FFH. 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 V_{REF} divided by 1024.

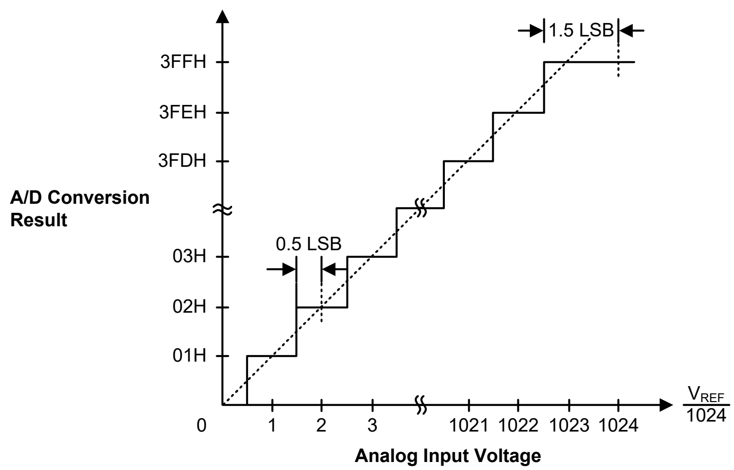
$$1\ LSB = V_{REF} \div 1024$$

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

$$A/D\ input\ voltage = A/D\ output\ digital\ value \times (V_{REF} \div 1024)$$

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 AV_{DD}.



Ideal A/D Transfer Function

A/D Conversion Programming Examples

The following two programming examples illustrate how to configure 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: using an ADBZ polling method to detect the end of conversion

```
clr  ADE             ; disable ADC interrupt
mov  a,03h           ; select fSYS/8 as A/D clock
mov  SADC1,a
mov  a,02h           ; set PASR to configure pin AN0
```

```
mov  PASR,a
mov  a,20h
mov  SADC0,a          ; enable A/D 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,SADOL          ; read low byte conversion result value
mov  SADOL_buffer,a   ; save result to user defined register
mov  a,SADOH          ; read high byte conversion result value
mov  SADOH_buffer,a   ; save result to user defined register
:
:
jmp  start_conversion ; start next A/D conversion
```

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

```
clr  ADE              ; disable ADC interrupt
mov  a,03h            ; select fSYS/8 as A/D clock
mov  SADC1,a
mov  a,02h            ; set PASR to configure pin AN0
mov  PASR,a
mov  a,20h
mov  SADC0,a          ; enable A/D 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_ISR:              ; ADC interrupt service routine
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, SADOL         ; read low byte conversion result value
mov  SADOL_buffer,a   ; save result to user defined register
mov  a, SADOH         ; read high byte conversion result value
mov  SADOH_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
```

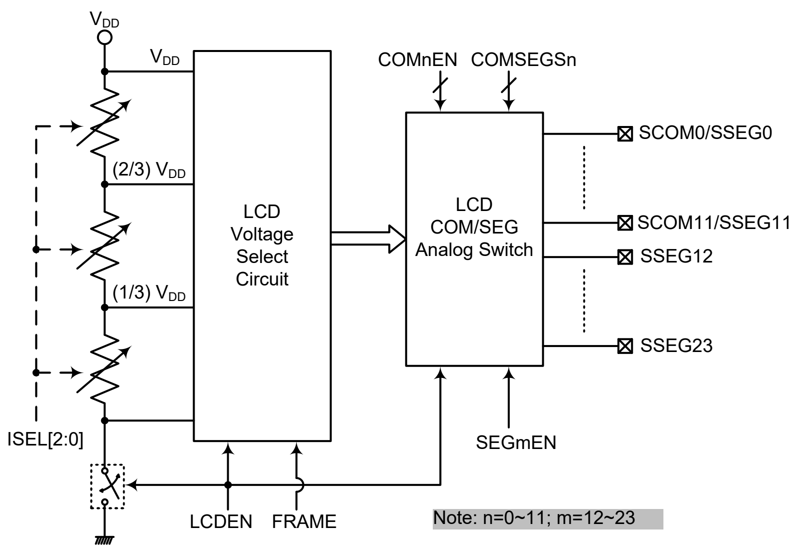
Software Controlled LCD Driver

The device has the capability of driving external LCD panels. The common and segment pins for LCD driving, SCOM0~SCOM11 and SSEG0~SSEG23, are pin-shared with certain functions on the I/O ports. The LCD signals, COM and SEG, 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 and segment pins. The LCD driver function is controlled using the LCD control registers which in addition to controlling the overall on/off function also controls the R-type bias current on the SCOM and SSEG pins. This enables the LCD COM and SEG driver to generate the necessary V_{SS}, (1/3)V_{DD}, (2/3)V_{DD} and V_{DD} voltage levels for LCD 1/3 bias operation.

The LCDEN bit in the SLCDC0 register is the overall master control for the LCD driver. The COMnEN, COMSEGSn and SEGmEN bits in the SLCDC1~SLCDC4 registers determine which pins on I/O Ports are used as LCD COM or SEG functions and which pins are not to be used as the LCD COM and SEG functions. When the pin is selected to be the SCOM or SSEG function, its original function whether it is an I/O or other pin-shared functions will be removed. 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 Frames

A cyclic LCD waveform includes two frames known as Frame 0 and Frame 1 for which the following offers a functional explanation.

Frame 0

To select Frame 0, clear the FRAME bit in the SLCDC0 register to 0.

In Frame 0, the COM signal output can have a value of V_{DD} or a V_{BIAS} value of $(1/3)\times V_{DD}$. The SEG signal output can have a value of V_{SS} or a V_{BIAS} value of $(2/3)\times V_{DD}$.

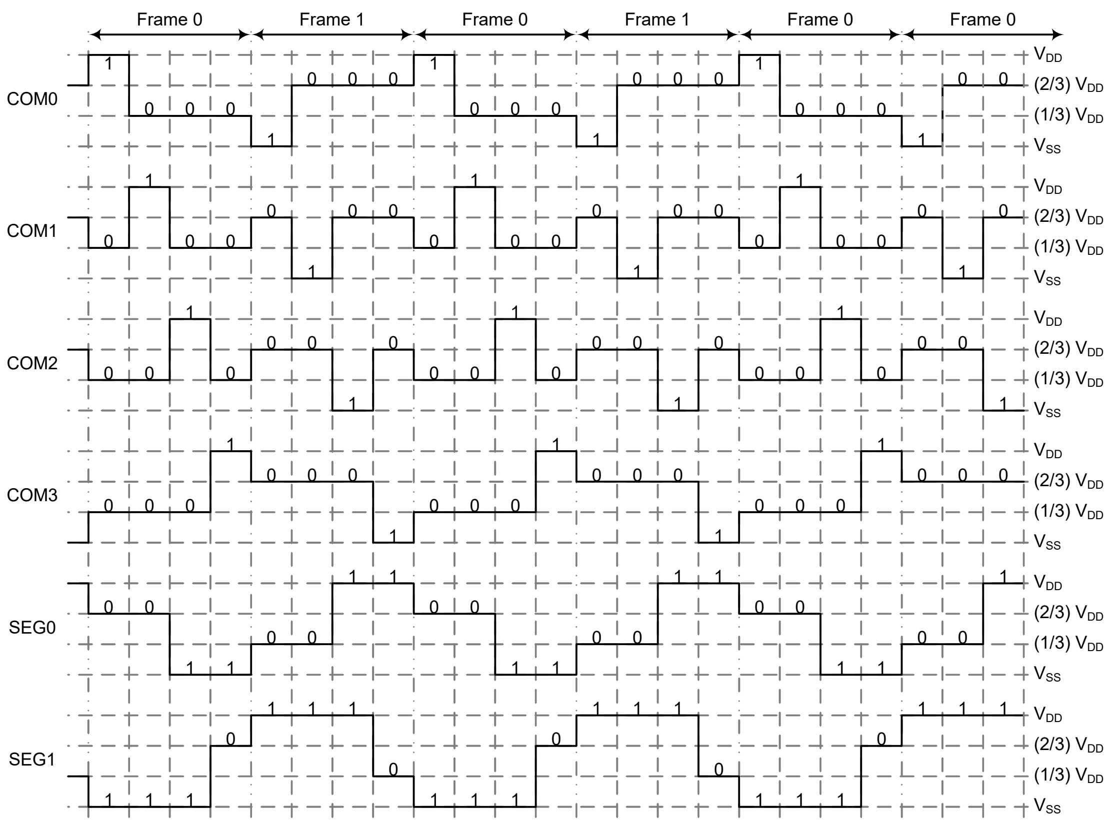
Frame 1

To select Frame 1, set the FRAME bit in the SLCDC0 register to 1.

In Frame 1, the COM signal output can have a value of V_{SS} or a V_{BIAS} value of $(2/3)\times V_{DD}$. The SEG signal output can have a value of V_{DD} or a V_{BIAS} value of $(1/3)\times V_{DD}$.

The COMn waveform is controlled by the application program using the FRAME bit in the SLCDC0 register and the corresponding pin-shared I/O data bit for the respective COM pin to determine whether the COMn output has a value of V_{DD} or $(1/3)V_{DD}$ in Frame 0 or $(2/3)V_{DD}$ or V_{SS} in Frame 1. The SEGm waveform is controlled in a similar way using the FRAME bit and the corresponding pin-shared I/O data bit for the respective SEG pin to determine whether the SEGm output has a value of V_{DD}, V_{SS} or V_{BIAS}.

The accompanying waveform diagram shows a typical 1/3 bias LCD waveform generated using the application program together with the LCD voltage select circuit. Note that the depiction of a “1” in the diagram illustrates an illuminated LCD pixel.



Note: The logical values shown in the above diagram are the corresponding pin-shared I/O data bit value.

1/3 Bias LCD Waveform – 4-COM & 2-SEG Application

LCD Control Registers

The LCD COM and SEG driver enables a range of selections to be provided to suit the requirement of the LCD panel which is being used. The bias resistor choice is implemented using the ISEL2~ISEL0 bits in the SLCDC0 register. All SCOM and SSEG pins are pin-shared with I/O pins and selected as SCOM and SSEG pins using the corresponding pin function selection bits in the SLCDCn registers respectively.

Register Name	Bit							
	7	6	5	4	3	2	1	0
SLCDC0	FRAME	ISEL2	ISEL1	ISEL0	—	—	—	LCDEN
SLCDC1	COM7EN	COM6EN	COM5EN	COM4EN	COM3EN	COM2EN	COM1EN	COM0EN
SLCDC2	—	—	—	—	COM11EN	COM10EN	COM9EN	COM8EN
SLCDC3	COMSEGS7	COMSEGS6	COMSEGS5	COMSEGS4	COMSEGS3	COMSEGS2	COMSEGS1	COMSEGS0
SLCDC4	SEG15EN	SEG14EN	SEG13EN	SEG12EN	COMSEGS11	COMSEGS10	COMSEGS9	COMSEGS8
SLCDC5	SEG23EN	SEG22EN	SEG21EN	SEG20EN	SEG19EN	SEG18EN	SEG17EN	SEG16EN

Software Controlled LCD Driver Register List

• SLCDC0 Register

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

Bit 7 **FRAME**: SCOM/SSEG Output Frame selection
0: Frame 0
1: Frame 1

Bit 6~4 **ISEL2~ISEL0**: R type LCD bias current selection (@V_{DD}=5V, 1/3 Bias)
000: $I_{BIAS=}$8.3μA, 3×200kΩ resistor selected
001: $I_{BIAS=}$16.6μA, 3×100kΩ resistor selected
010: $I_{BIAS=}$50μA, 3×33.3kΩ resistor selected
011: $I_{BIAS=}$100μA, 3×16.6kΩ resistor selected
100: $I_{BIAS=}$500μA, 3×3.33kΩ resistor selected
101: $I_{BIAS=}$1000μA, 3×1.67kΩ resistor selected
110: $I_{BIAS=}$2000μA, 3×833Ω resistor selected
111: $I_{BIAS=}$4000μA, 3×417Ω resistor selected

Bit 3~1 Unimplemented, read as "0"

Bit 0 **LCDEN**: LCD function control
0: Disable
1: Enable
The SCOMn and SSEGm lines can be enabled using COMnEN and SEGmEN if the LCDEN bit is set to 1. When the LCDEN bit is cleared to 0, then the SCOMn and SSEGm outputs will be fixed at a V_{SS} level.

• SLCDC1 Register

Bit	7	6	5	4	3	2	1	0
Name	COM7EN	COM6EN	COM5EN	COM4EN	COM3EN	COM2EN	COM1EN	COM0EN
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 **COM7EN**: SCOM7/SSEG7 or other pin function selection
0: Other pin-shared functions
1: SCOM7/SSEG7 function

Bit 6 **COM6EN**: SCOM6/SSEG6 or other pin function selection
0: Other pin-shared functions
1: SCOM6/SSEG6 function

Bit 5 **COM5EN**: SCOM5/SSEG5 or other pin function selection
0: Other pin-shared functions
1: SCOM5/SSEG5 function

Bit 4 **COM4EN**: SCOM4/SSEG4 or other pin function selection
0: Other pin-shared functions
1: SCOM4/SSEG4 function

Bit 3 **COM3EN**: SCOM3/SSEG3 or other pin function selection
0: Other pin-shared functions
1: SCOM3/SSEG3 function

Bit 2 **COM2EN**: SCOM2/SSEG2 or other pin function selection
0: Other pin-shared functions
1: SCOM2/SSEG2 function

Bit 1 **COM1EN**: SCOM1/SSEG1 or other pin function selection
0: Other pin-shared functions
1: SCOM1/SSEG1 function

Bit 0 **COM0EN**: SCOM0/SSEG0 or other pin function selection
0: Other pin-shared functions
1: SCOM0/SSEG0 function

- **SLCDC2 Register**

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	COM11EN	COM10EN	COM9EN	COM8EN
R/W	—	—	—	—	R/W	R/W	R/W	R/W
POR	—	—	—	—	0	0	0	0

Bit 7~4 Unimplemented, read as "0"

Bit 3 **COM11EN**: SCOM11/SSEG11 or other pin function selection
0: Other pin-shared functions
1: SCOM11/SSEG11 function

Bit 2 **COM10EN**: SCOM10/SSEG10 or other pin function selection
0: Other pin-shared functions
1: SCOM10/SSEG10 function

Bit 1 **COM9EN**: SCOM9/SSEG9 or other pin function selection
0: Other pin-shared functions
1: SCOM9/SSEG9 function

Bit 0 **COM8EN**: SCOM8/SSEG8 or other pin function selection
0: Other pin-shared functions
1: SCOM8/SSEG8 function

- **SLCDC3 Register**

Bit	7	6	5	4	3	2	1	0
Name	COMSEGS7	COMSEGS6	COMSEGS5	COMSEGS4	COMSEGS3	COMSEGS2	COMSEGS1	COMSEGS0
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 **COMSEGS7**: SCOM7 or SSEG7 pin function selection
0: SCOM7
1: SSEG7

Bit 6 **COMSEGS4**: SCOM6 or SSEG6 pin function selection
0: SCOM6
1: SSEG6

Bit 5 **COMSEGS5**: SCOM5 or SSEG5 pin function selection
0: SCOM5
1: SSEG5

Bit 4 **COMSEGS4**: SCOM4 or SSEG4 pin function selection
0: SCOM4
1: SSEG4

Bit 3 **COMSEGS3**: SCOM3 or SSEG3 pin function selection
0: SCOM3
1: SSEG3

Bit 2 **COMSEGS2**: SCOM2 or SSEG2 pin function selection
0: SCOM2
1: SSEG2

Bit 1 **COMSEGS1**: SCOM1 or SSEG1 pin function selection
0: SCOM1
1: SSEG1

Bit 0 **COMSEGS0**: SCOM0 or SSEG0 pin function selection
0: SCOM0
1: SSEG0

• SLCDC4 Register

Bit	7	6	5	4	3	2	1	0
Name	SEG15EN	SEG14EN	SEG13EN	SEG12EN	COMSEGS11	COMSEGS10	COMSEGS9	COMSEGS8
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 **SEG15EN**: SSEG15 pin function selection
0: Other pin-shared functions
1: SSEG15 function

Bit 6 **SEG14EN**: SSEG14 pin function selection
0: Other pin-shared functions
1: SSEG14 function

Bit 5 **SEG13EN**: SSEG13 pin function selection
0: Other pin-shared functions
1: SSEG13 function

Bit 4 **SEG12EN**: SSEG12 pin function selection
0: Other pin-shared functions
1: SSEG12 function

Bit 3 **COMSEGS11**: SCOM11 or SSEG11 pin function selection
0: SCOM11
1: SSEG11

Bit 2 **COMSEGS10**: SCOM10 or SSEG10 pin function selection
0: SCOM10
1: SSEG10

Bit 1 **COMSEGS9**: SCOM9 or SSEG9 pin function selection
0: SCOM9
1: SSEG9

Bit 0 **COMSEGS8**: SCOM8 or SSEG8 pin function selection
0: SCOM8
1: SSEG8

• SLCDC5 Register

Bit	7	6	5	4	3	2	1	0
Name	SEG23EN	SEG22EN	SEG21EN	SEG20EN	SEG19EN	SEG18EN	SEG17EN	SEG16EN
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 **SEG23EN**: SSEG23 pin function selection
0: Other pin-shared functions
1: SSEG23 function

Bit 6	**SEG22EN**: SSEG22 pin function selection 0: Other pin-shared functions 1: SSEG22 function
Bit 5	**SEG21EN**: SSEG21 pin function selection 0: Other pin-shared functions 1: SSEG21 function
Bit 4	**SEG20EN**: SSEG20 pin function selection 0: Other pin-shared functions 1: SSEG20 function
Bit 3	**SEG19EN**: SSEG19 pin function selection 0: Other pin-shared functions 1: SSEG19 function
Bit 2	**SEG18EN**: SSEG18 pin function selection 0: Other pin-shared functions 1: SSEG18 function
Bit 1	**SEG17EN**: SSEG17 pin function selection 0: Other pin-shared functions 1: SSEG17 function
Bit 0	**SEG16EN**: SSEG16 pin function selection 0: Other pin-shared functions 1: SSEG16 function

Interrupts

Interrupts are an important part of any microcontroller system. When an external event or an internal function such as a Timer Module or an A/D converter requires microcontroller attention, their corresponding interrupt will enforce a temporary suspension of the main program allowing the microcontroller to direct attention to their respective needs. The device contains several external interrupt and internal interrupt functions. The external interrupts are generated by the action of the external INT pin, while the internal interrupts are generated by various internal functions such as TMs and the A/D converter, etc.

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 registers fall into three categories. The first is the INTC0~INTC1 registers which configure the primary interrupts, the second is the MFI0~MFI1 registers which configures the Multi-function interrupts. Finally there is an INTEG register to setup the external interrupt trigger edge type.

Each 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	—	—
INT Pin	INTE	INTF	—
Multi-function	MFnE	MFnF	n=0~1
A/D Converter	ADE	ADF	—
CTM	CTMnPE	CTMnPF	n=0~1
	CTMnAE	CTMnAF	
STM	STMPE	STMPF	—
	STMAE	STMAF	—

Interrupt Register Bit Naming Conventions

Register Name	Bit							
	7	6	5	4	3	2	1	0
INTEG	—	—	—	—	—	—	INTS1	INTS0
INTC0	—	MF0F	—	INTF	MF0E	—	INTE	EMI
INTC1	—	ADF	—	MF1F	—	ADE	—	MF1E
MFI0	—	—	STMAF	STMPF	—	—	STMAE	STMPE
MFI1	CTM1AF	CTM1PF	CTM0AF	CTM0PF	CTM1AE	CTM1PE	CTM0AE	CTM0PE

Interrupt Register List

• INTEG Register

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	—	—	INTS1	INTS0
R/W	—	—	—	—	—	—	R/W	R/W
POR	—	—	—	—	—	—	0	0

Bit 7~2 Unimplemented, read as "0"

Bit 1~0 **INTS1~INTS0**: Interrupt edge control for INT 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	—	MF0F	—	INTF	MF0E	—	INTE	EMI
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 **MF0F**: Multi-function 0 interrupt request flag
0: No request
1: Interrupt request

Bit 5 Unimplemented, read as "0"

Bit 4 **INTF**: INT interrupt request flag
0: No request
1: Interrupt request

Bit 3 **MF0E**: Multi-function 0 interrupt control
0: Disable
1: Enable

Bit 2	Unimplemented, read as "0"

Bit 1	**INTE**: INT 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	—	ADF	—	MF1F	—	ADE	—	MF1E
R/W	—	R/W	—	R/W	—	R/W	—	R/W
POR	—	0	—	0	—	0	—	0

Bit 7	Unimplemented, read as "0"

Bit 6	**ADF**: A/D Converter interrupt request flag
0: No request
1: Interrupt request

Bit 5	Unimplemented, read as "0"

Bit 4	**MF1F**: Multi-function 1 interrupt request flag
0: No request
1: Interrupt request

Bit 3	Unimplemented, read as "0"

Bit 2	**ADE**: A/D Converter interrupt control
0: Disable
1: Enable

Bit 1	Unimplemented, read as "0"

Bit 0	**MF1E**: Multi-function 1 interrupt control
0: Disable
1: Enable

• MFI0 Register

Bit	7	6	5	4	3	2	1	0
Name	—	—	STMAF	STMPF	—	—	STMAE	STMPE
R/W	—	—	R/W	R/W	—	—	R/W	R/W
POR	—	—	0	0	—	—	0	0

Bit 7~6	Unimplemented, read as "0"

Bit 5	**STMAF**: STM Comparator A match interrupt request flag
0: No request
1: Interrupt request

Bit 4	**STMPF**: STM Comparator P match interrupt request flag
0: No request
1: Interrupt request

Bit 3~2	Unimplemented, read as "0"

Bit 1	**STMAE**: STM Comparator A match interrupt control
0: Disable
1: Enable

Bit 0	**STMPE**: STM Comparator P match interrupt control
0: Disable
1: Enable

• **MFI1 Register**

Bit	7	6	5	4	3	2	1	0
Name	CTM1AF	CTM1PF	CTM0AF	CTM0PF	CTM1AE	CTM1PE	CTM0AE	CTM0PE
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 **CTM1AF**: CTM1 Comparator A match interrupt request flag
0: No request
1: Interrupt request

Bit 6 **CTM1PF**: CTM1 Comparator P match interrupt request flag
0: No request
1: Interrupt request

Bit 5 **CTM0AF**: CTM0 Comparator A match interrupt request flag
0: No request
1: Interrupt request

Bit 4 **CTM0PF**: CTM0 Comparator P match interrupt request flag
0: No request
1: Interrupt request

Bit 3 **CTM1AE**: CTM1 Comparator A match interrupt control
0: Disable
1: Enable

Bit 2 **CTM1PE**: CTM1 Comparator P match interrupt control
0: Disable
1: Enable

Bit 1 **CTM0AE**: CTM0 Comparator A match interrupt control
0: Disable
1: Enable

Bit 0 **CTM0PE**: CTM0 Comparator P match interrupt control
0: Disable
1: Enable

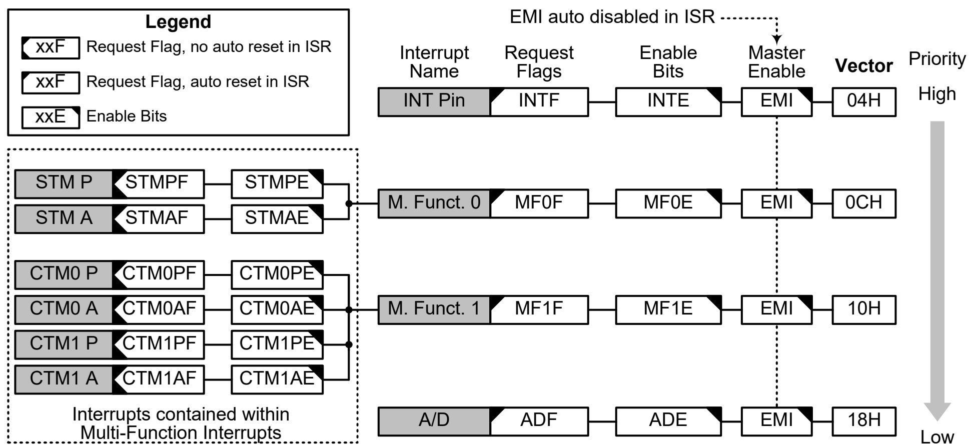
Interrupt Operation

When the conditions for an interrupt event occur, such as a TM Comparator P or Comparator A match etc., 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.

The various interrupt enable bits, together with their associated request flags, are shown in the Accompanying diagrams with their order of priority. Some interrupt sources have their own individual vector while others share the same multi-function interrupt vector. 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 accompanying 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.



Interrupt Structure

External Interrupt

The external interrupt is controlled by signal transitions on the INT pin. An external interrupt request will take place when the external interrupt request flag, INTF, 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, INTE, 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, INTF, 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.

Multi-function Interrupts

Within the device there are two Multi-function interrupts. Unlike the other independent interrupts, these interrupts have no independent source, but rather are formed from other existing interrupt sources, namely the TM interrupts.

A Multi-function interrupt request will take place when any of the Multi-function interrupt request flags MFnF are set. The Multi-function interrupt flag will be set when any of their included functions generate an interrupt request flag. To allow the program to branch to its respective interrupt vector address, when the Multi-function interrupt is enabled and the stack is not full, and any one of the interrupts contained within each of the Multi-function interrupt occurs, a subroutine call to the related Multi-function interrupt vector will take place. When the interrupt is serviced, the related Multi-function request flag will be automatically reset and the EMI bit will be automatically cleared to disable other interrupts.

However, it must be noted that, although the Multi-function Interrupt request flag will be automatically reset when the interrupt is serviced, the request flag from the original source of the Multi-function interrupt will not be automatically reset and must be manually reset by the application program.

A/D Converter Interrupt

The A/D Converter Interrupt is controlled by the termination of an A/D conversion process. 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 Converter Interrupt vector, will take place. When the interrupt is serviced, the A/D Converter Interrupt flag, ADF, will be automatically cleared. The EMI bit will also be automatically cleared to disable other interrupts.

Timer Module Interrupts

The Compact and Standard TMs each have two interrupts, one comes from the comparator A match situation and the other comes from the comparator P match situation. All of the TM interrupts are contained within the Multi-function Interrupts. For all of the TM types there are two interrupt request flags and two enable control bits. A TM interrupt request will take place when any of the TM request flags are set, a situation which occurs when a TM comparator P or A match situation happens.

To allow the program to branch to its respective interrupt vector address, the global interrupt enable bit, EMI, respective TM Interrupt enable bit, and relevant Multi-function Interrupt enable bit, MFnE, must first be set. When the interrupt is enabled, the stack is not full and a TM comparator match situation occurs, a subroutine call to the relevant Multi-function Interrupt vector locations, will take place. When the TM interrupt is serviced, the EMI bit will be automatically cleared to disable other interrupts. However, only the related MFnF flag will be automatically cleared. As the TM interrupt request flags will not be automatically cleared, they have to be cleared by the application program.

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.

Where a certain interrupt is contained within a Multi-function interrupt, then when the interrupt service routine is executed, as only the Multi-function interrupt request flags, MFnF, will be automatically cleared, the individual request flag for the function needs to be cleared by the application program.

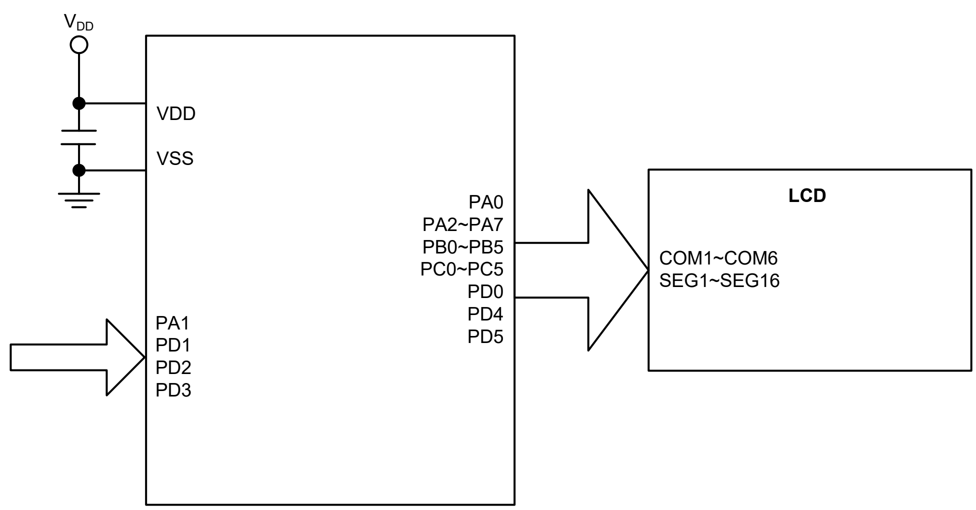
It is recommended that programs do not use the "CALL" instruction within the interrupt service subroutine. Interrupts often occur in an unpredictable manner or need to be serviced immediately. If only one stack is left and the interrupt is not well controlled, the original control sequence will be damaged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

Every interrupt has the capability of waking up the microcontroller when it is in 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.

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 Holtek 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 Holtek 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 Holtek 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 Holtek 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, Holtek 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	1[Note]	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	1[Note]	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	1[Note]	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	1[Note]	Z, C, AC, OV
DAA [m]	Decimal adjust ACC for Addition with result in Data Memory	1[Note]	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	1[Note]	Z
ORM A,[m]	Logical OR ACC to Data Memory	1[Note]	Z
XORM A,[m]	Logical XOR ACC to Data Memory	1[Note]	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	1[Note]	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	1[Note]	Z
DECA [m]	Decrement Data Memory with result in ACC	1	Z
DEC [m]	Decrement Data Memory	1[Note]	Z
Rotate			
RRA [m]	Rotate Data Memory right with result in ACC	1	None
RR [m]	Rotate Data Memory right	1[Note]	None
RRCA [m]	Rotate Data Memory right through Carry with result in ACC	1	C
RRC [m]	Rotate Data Memory right through Carry	1[Note]	C
RLA [m]	Rotate Data Memory left with result in ACC	1	None
RL [m]	Rotate Data Memory left	1[Note]	None
RLCA [m]	Rotate Data Memory left through Carry with result in ACC	1	C
RLC [m]	Rotate Data Memory left through Carry	1[Note]	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
CLR WDT1	Pre-clear Watchdog Timer	1	TO, PDF
CLR WDT2	Pre-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 up to 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.

3. For the "CLR WDT1" and "CLR WDT2" instructions the TO and PDF flags may be affected by the execution status. The TO and PDF flags are cleared after both "CLR WDT1" and "CLR WDT2" instructions are consecutively executed. Otherwise the TO and PDF flags remain unchanged.

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

CLR WDT1 Pre-clear Watchdog Timer

Description: The TO, PDF flags and the WDT are all cleared. Note that this instruction works in conjunction with CLR WDT2 and must be executed alternately with CLR WDT2 to have effect. Repetitively executing this instruction without alternately executing CLR WDT2 will have no effect.

Operation: WDT cleared
TO ← 0
PDF ← 0

Affected flag(s): TO, PDF

CLR WDT2 Pre-clear Watchdog Timer

Description: The TO, PDF flags and the WDT are all cleared. Note that this instruction works in conjunction with CLR WDT1 and must be executed alternately with CLR WDT1 to have effect. Repetitively executing this instruction without alternately executing CLR WDT1 will have no effect.

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: $[m] \leftarrow \overline{[m]}$

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: $ACC \leftarrow \overline{[m]}$

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: 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

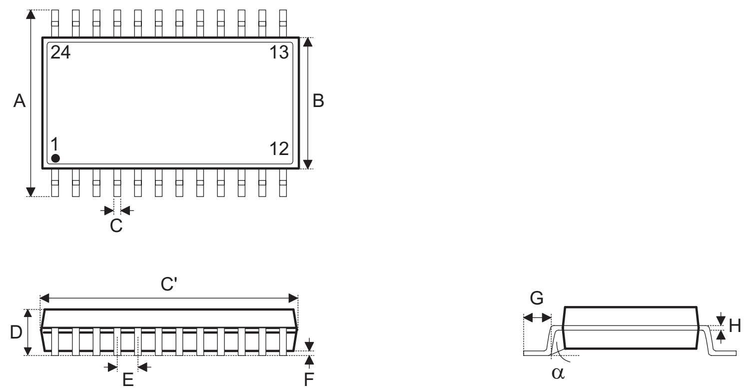
Package Information

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 Holtek 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

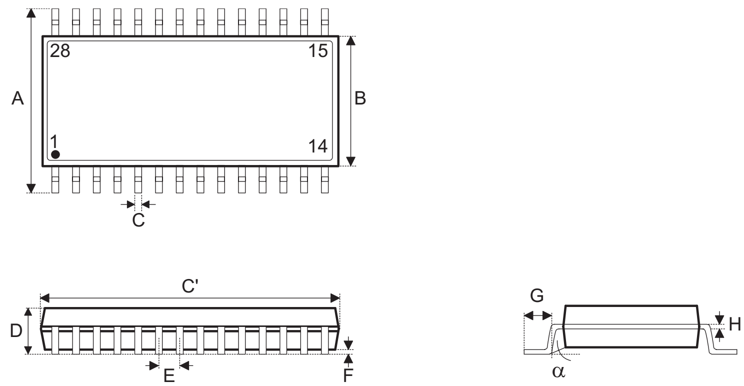
24-pin SOP (300mil) Outline Dimensions



Symbol	Dimensions in inch		
	Min.	Nom.	Max.
A	—	0.406 BSC	—
B	—	0.295 BSC	—
C	0.012	—	0.020
C'	—	0.606 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'	—	15.40 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°

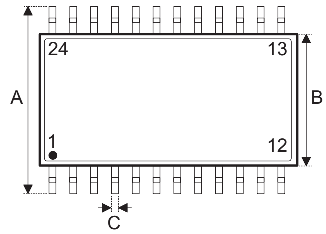
28-pin SOP (300mil) Outline Dimensions

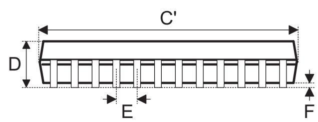


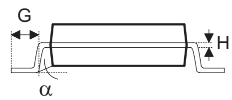
Symbol	Dimensions in inch		
	Min.	Nom.	Max.
A	—	0.406 BSC	—
B	—	0.295 BSC	—
C	0.012	—	0.020
C’	—	0.705 BSC	—
D	—	—	0.104
E	—	0.050 BSC	—
F	0.004	—	0.012
G	0.016	—	0.050
H	0.008	—	0.013
α	0°	—	8°

Symbol	Dimensions in mm		
	Min.	Nom.	Max.
A	—	10.300 BSC	—
B	—	7.500 BSC	—
C	0.31	—	0.51
C’	—	17.900 BSC	—
D	—	—	2.65
E	—	1.270 BSC	—
F	0.10	—	0.30
G	0.40	—	1.27
H	0.20	—	0.33
α	0°	—	8°

24-pin SSOP (150mil) Outline Dimensions



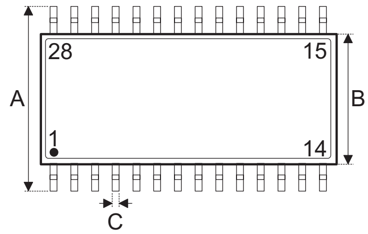


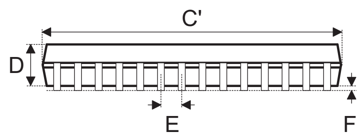


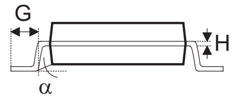
Symbol	Dimensions in inch		
	Min.	Nom.	Max.
A	—	0.236 BSC	—
B	—	0.154 BSC	—
C	0.008	—	0.012
C’	—	0.341 BSC	—
D	—	—	0.069
E	—	0.025 BSC	—
F	0.004	—	0.010
G	0.016	—	0.050
H	0.004	—	0.010
α	0°	—	8°

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°

28-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’	—	0.390 BSC	—
D	—	—	0.069
E	—	0.025 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	0.20	—	0.30
C’	—	9.90 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°