Charger Flash MCU

HT45F5Q-1

Revision: V1.02 Date: November 24, 2021

www.holtek.com

Table of Contents

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
- Oscillator type
  - Internal High Speed 8MHz RC – HIRC
- Fully integrated internal oscillator requires no external components
- FAST operation mode
- All instructions executed in one or two instruction cycles
- Table read instructions
- 61 powerful instructions
- 4-level subroutine nesting
- Bit manipulation instruction

Peripheral Features

- Flash Program Memory: 1K×14
- Data Memory: 32×8
- Emulated EEPROM Memory: 32×14 水 Watchdog Timer function
- 9 bidirectional I/O lines
- Single pin-shared external interrupt
- Dual Time-Base functions for generation of fixed time interrupt signals
- 5 external channel 10-bit resolution A/D converter with internal reference voltage V_{VR}
- Battery charge circuit
  - Two Operational Amplifier
  - One 8-bit D/A converter
  - One 12-bit D/A converter
- Low voltage reset function – LVR
- Package type: 16-pin NSOP

Development Tools

For rapid product development and to simplify device parameter setting, Holtek has provided relevant development tools which users can download from the following link:

https://www.holtek.com/ht45f5q-x-charger-development-workshop

https://www.holtek.com/ht45f5q-x-charger-volume-production-fixture

General Description

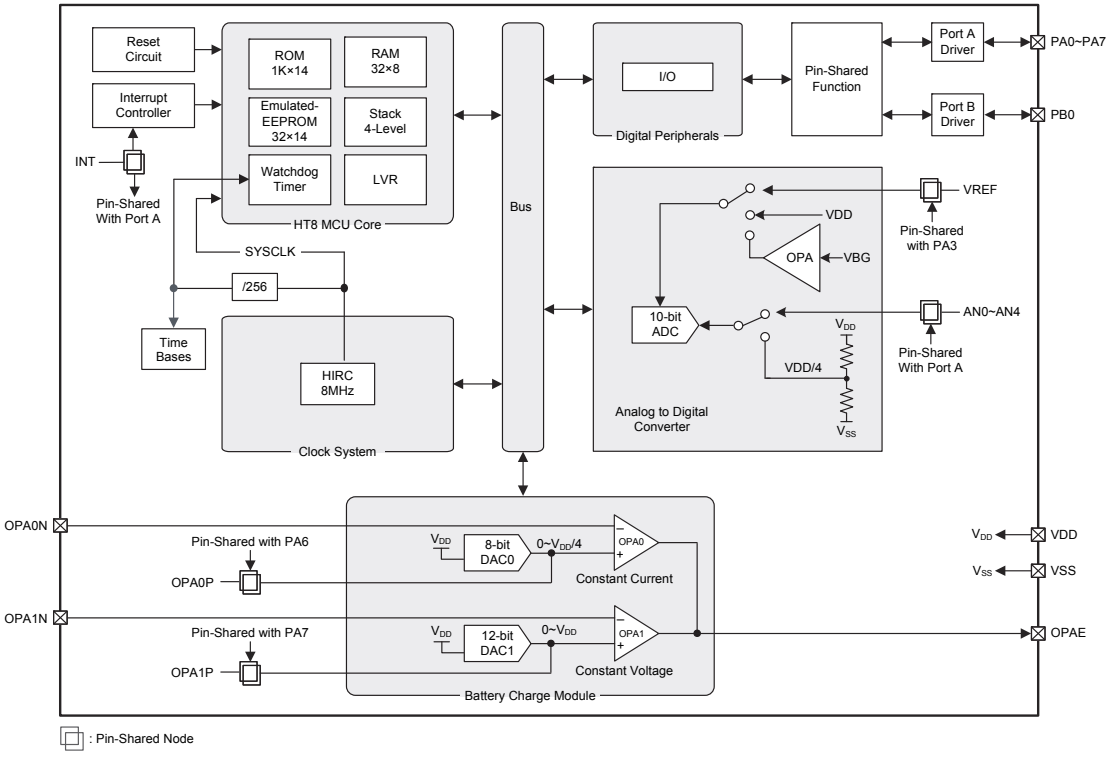
The HT45F5Q-1 is the second generation low-end charger Flash MCU specifically designed for Battery Charger applications. Offering users the convenience of Flash Memory multi-programming features, the device also includes a wide range of functions and features. Other memory includes an area of RAM Data 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 function, two operational amplifiers namely OPA0 and OPA1, which are specifically for constant current (CC) and constant voltage (CV) control respectively. Protective features such as an internal Watchdog Timer and low voltage reset function coupled with excellent noise immunity and ESD protection ensure that reliable operation is maintained in hostile electrical environments.

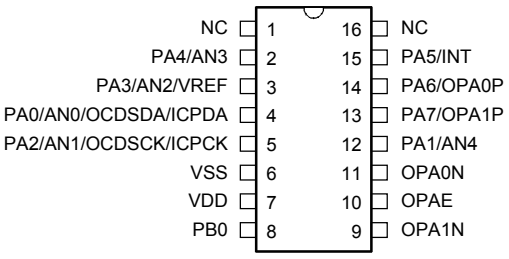
The device also includes fully integrated high speed oscillator which require no external components for its implementation.

The inclusion of flexible I/O programming features, Time-Base functions along with many other features ensure that the device will find excellent use in charger applications.

Block Diagram



Pin Assignment



HT45F5Q-1/HT45V5Q-1
16 NSOP-A

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

2. The OCDSDA and OCDSCK pins are supplied as OCDS dedicated pins and as such only available for the HT45V5Q-1 device which is the OCDS EV chip for the HT45F5Q-1 device.

Pin Description

With the exception of the power pins and some OPA pins, all pins on the device can be referenced by their Port name, e.g. PA0, PA1 etc., which refer to the digital I/O function of the pins. However these Port pins are also shared with other function such as the Analog to Digital Converter, external interrupt pins etc. 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.

Pin Name	Function	OPT	I/T	O/T	Description
PA0/AN0/OCDSDA/ICPDA	PA0	PAPU PAS0	ST	CMOS	General purpose I/O. Register enabled pull-up
	AN0	PAS0	AN	—	A/D converter external input channel 0
	OCDSDA	—	ST	CMOS	OCDS address/data, for EV chip only
	ICPDA	—	ST	CMOS	ICP address/data
PA1/AN4	PA1	PAPU PAS0	ST	CMOS	General purpose I/O. Register enabled pull-up
	AN4	PAS0	AN	—	A/D Converter external input channel 4
PA2/AN1/OCDSCK/ICPCK	PA2	PAPU PAS0	ST	CMOS	General purpose I/O. Register enabled pull-up
	AN1	PAS0	AN	—	A/D Converter external input channel 1
	OCDSCK	—	ST	—	OCDS clock pin, for EV chip only
	ICPCK	—	ST	—	ICP clock pin
PA3/AN2/VREF	PA3	PAPU PAS0	ST	CMOS	General purpose I/O. Register enabled pull-up
	AN2	PAS0	AN	—	A/D Converter external input channel 2
	VREF	PAS0	AN	—	A/D Converter reference voltage input
PA4/AN3	PA4	PAPU PAS1	ST	CMOS	General purpose I/O. Register enabled pull-up
	AN3	PAS1	AN	—	A/D Converter external input channel 3
PA5/INT	PA5	PAPU	ST	CMOS	General purpose I/O. Register enabled pull-up
	INT	INTEG INTC0	ST	—	External Interrupt input
PA6/OPA0P	PA6	PAPU PAS1	ST	CMOS	General purpose I/O. Register enabled pull-up
	OPA0P	PAS1	AN	—	Positive input of OPA0

Pin Name	Function	OPT	I/T	O/T	Description
PA7/OPA1P	PA7	PAPU PAS1	ST	CMOS	General purpose I/O. Register enabled pull-up
	OPA1P	—	AN	—	Positive input of OPA1
PB0	PB0	—	ST	CMOS	General purpose I/O. Register enabled pull-up
OPA0N	OPA0N	—	AN	—	Negative input of OPA0
OPA1N	OPA1N	—	AN	—	Negative input of OPA1
OPAE	OPAE	—	—	AN	Analog output of OPA
VDD	VDD	—	PWR	—	Digital and analog positive power supply
VSS	VSS	—	PWR	—	Digital and analog negative power supply, ground

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

Absolute Maximum Ratings

Supply Voltage ..... V_{SS}−0.3V to 6.0V
Input Voltage ..... V_{SS}−0.3V to V_{DD}+0.3V
Storage Temperature ..... -50°C to 125°C
Operating Temperature ..... -40°C to 85°C
I_{OH} Total ..... -80mA
I_{OL} Total 80mA
Total Power Dissipation 500mW

Note: These are stress ratings only. Stresses exceeding the range specified under “Absolute Maximum Ratings” may cause substantial damage to the device. Functional operation of the devices 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, 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

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	3V	f_{SYS}=8MHz, OPA0/OPA1 enable	—	1.1	2.0	mA
		5V		—	2.2	3.6	mA

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

1. Any digital inputs are setup 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

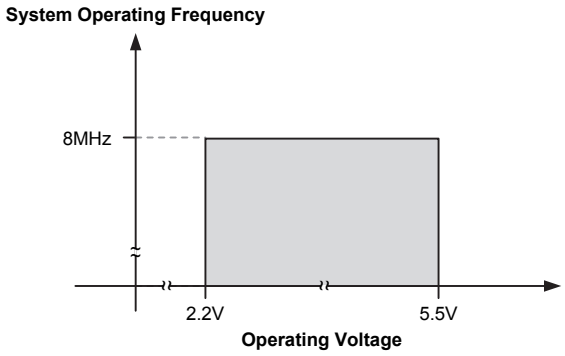
For data in the following tables, note that factors such as operating voltage and temperature, etc., can all exert an influence on the measured values.

High Speed Internal Oscillator – HIRC – Frequency Accuracy

During the program writing operation the writer will trim the HIRC oscillator at 8MHz frequency and user selected voltage of 5V.

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Temp.				
f_{HIRC}	High Speed Internal HIRC Frequency	5V	Ta=25°C	-2%	8	+ 2%	MHz
		2.2V~5.5V	Ta=25°C	-10%	8	+10%	
		5V	Ta=0°C~70°C	-3%	8	+3%	
		5V	Ta=-40°C~85°C	-5%	8	+5%	
		2.2V~5.5V	Ta=0°C~70°C	-10%	8	+10%	
		2.2V~5.5V	Ta=-40°C~85°C	-15%	8	+15%	

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_{RSTD}	System Reset Delay Time Reset Source from Power-on Reset or LVR Hardware Reset	—	RR_{POR}=5V/ms	14	16	20	ms
	System Reset Delay Time LVRC/WDTC Reset	—	—	14	16	20	
	System Reset Delay Time Reset Source from WDT Overflow	—	—	14	16	20	
t_{SRESET}	Minimum Software Reset Width to Reset	—	—	45	90	120	μs

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 or Input Pins	5V	—	0	—	1.5	V
		—		0	—	$0.2V_{DD}$	
V_{IH}	Input High Voltage for I/O Ports or Input Pins	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 Rsistance 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_{INT}	External Interrupt Minimum Pulse Width	—	—	0.3	—	—	μs

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

A/D Converter Electrical Characteristics

Ta=-40°C~85°C

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
V_{DD}	Operating Voltage	—	—	2.2	—	5.5	V
V_{ADI}	A/D Converter Input Voltage	—	—	0	—	V_{REF}	V
V_{REF}	A/D Converter Reference Voltage	—	—	2.2	—	V_{DD}	V
N_R	Resolution	—	—	—	—	10	Bit
DNL	Differential Non-linearity	—	V_{REF}=V_{DD}, t_{ADCK}=0.5μs	-1.5	—	+1.5	LSB
INL	Integral Non-linearity	—	V_{REF}=V_{DD}, t_{ADCK}=0.5μs	-2	—	+2	LSB
I_{ADC}	Additional Current for A/D Converter Enable	2.2V	No load, t_{ADCK}=0.5μs	—	300	420	μA
		3V		—	340	500	
		5V		—	500	700	
t_{ADCK}	A/D Clock Period	—	—	0.5	—	10	μs
t_{ON2ST}	A/D Converter On-to-Start Time	—	—	4	—	—	μs
t_{ADS}	A/D Sampling Time	—	—	—	4	—	t_{ADCK}
t_{ADC}	A/D Conversion Time (Include A/D Sample and Hold Time)	—	—	—	14	—	t_{ADCK}
GERR	A/D Conversion Gain Error	—	V_{REF}=V_{DD}	-2	—	2	LSB
OSRR	A/D Conversion Offset Error	—	V_{REF}=V_{DD}	-2	—	2	LSB
I_{OPA}	Additional Current for OPA Enable	3V	No load	—	390	550	μA
		5V	No load	—	500	650	
V_{OR}	OPA Maximum Output Voltage Range	3V	—	V_{SS}+0.1	—	V_{DD}–0.1	V
		5V	—	V_{SS}+0.1	—	V_{DD}–0.1	
V_{VR}	Fix Voltage Output of OPA	2.2V~5.5V	V_{RI}=V_{BG}=1.03V±20%	-5%	1.6	+5%	V

D/A Converter Electrical Characteristics

Ta=-40°C~85°C

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
V_{DD}	Operating Voltage	—	—	2.2	—	5.5	V
V_{DAC}	D/A Converter 0 Output Voltage Range	—	—	V_{SS}	—	V_{DD}/4	V
	D/A Converter 1 Output Voltage Range	—	—	V_{SS}	—	V_{DD}	V
I_{DAC}	Additional Current for D/A Converter 0 Enable	5V	—	—	500	600	μA
	Additional Current for D/A Converter 1 Enable	5V	—	—	500	600	μA
t_{ST}	Settling Time	5V	C_{LOAD}=50pF	—	—	5	μs
DNL	Differential Nonlinearity for D/A Converter 0	5V	V_{REF}=V_{DD}	—	—	±1	LSB
	Differential Nonlinearity for D/A Converter 1	5V	V_{REF}=V_{DD}	—	±4	±10	LSB
INL	Integral Nonlinearity for D/A Converter 0	5V	V_{REF}=V_{DD}	—	—	±1.5	LSB
	Integral Nonlinearity for D/A Converter 1	5V	V_{REF}=V_{DD}	—	±6	±12	LSB
Ro	DAC0 R2R Output Resistor	5V	—	—	10	—	kΩ
	DAC1 R2R Output Resistor	5V	—	—	13	—	kΩ

Operational Amplifier Electrical Characteristics

Ta=-40°C~85°C

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
I_{OPA}	Additional Current for Each OPA	5V	No load	—	300	600	μA
V_{OS}	Input Offset Voltage	5V	Without calibration	-15	—	15	mV
V_{CM}	Common Mode Voltage Range	5V	—	V_{SS}	—	V_{DD}-1.4	V
V_{OR}	Maximum Output Voltage Range	5V	—	V_{SS}+0.1	—	V_{DD}-0.1	V
SR	Slew Rate	5V	No load	0.6	1.8	—	V/μs
GBW	Gain Bandwidth	5V	R_{LOAD}=1MΩ, C_{LOAD}=100pF	1.0	1.8	—	MHz
PSRR	Power Supply Rejection Ratio	5V	—	60	80	—	dB
CMRR	Common Mode Rejection Ratio	5V	—	60	80	—	dB
I_{SINK}	Output Sink Current	3V	V_{OL}=0.3V	0.7	1.2	—	mA
		5V	V_{OL}=0.5V	1.6	2.8	—	mA

Memory Characteristics

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

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
Flash Program Memory / Emulated EEPROM Memory							
V_{DD}	Operating Voltage for Read	—	—	2.2	—	5.5	V
	Operating Voltage for Erase/Write	—	—	4.5	5.0	5.5	
t_{DEW}	Erase/Write Time – Flash Program Memory	5V	Ta=25°C	—	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	
E_P	Cell Endurance	—	—	10K	—	—	E/W
t_{RETD}	ROM Data Retention Time	—	Ta=25°C	—	40	—	Year

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
RAM Data Memory							
V_{DD}	Operating Voltage for Read/Write	—	—	V_{DDmin}	—	V_{DDmax}	V
V_{DR}	RAM Data Retention Voltage	—	—	1	—	—	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 is 2.1V	-5%	2.1	+5%	V
I_{LVRBG}	Operating Current	3V	LVR enable, V_{LVR}=2.1V	—	—	15	μA
		5V		—	15	25	
I_{LVR}	Additional Current for LVR Enable	5V	—	—	—	25	μA
t_{LVR}	Minimum Low Voltage Width to Reset	—	—	120	240	480	μs

Internal Reference Voltage Characteristics

Ta=25°C

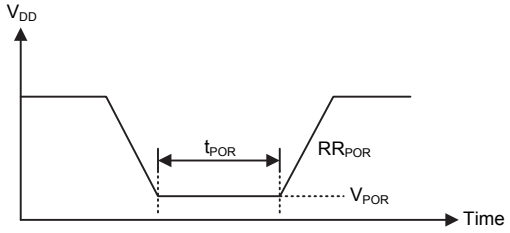
Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
I_{BG}	Additional Current for Bandgap Reference Enable	—	VBGEN=1, LVR disable	—	—	2	μA
t_{BGS}	V_{BG} Turn-on Stable Time	—	No load	—	—	50	μs

Note: The V_{BG} voltage is used as the A/D converter internal OPA input.

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



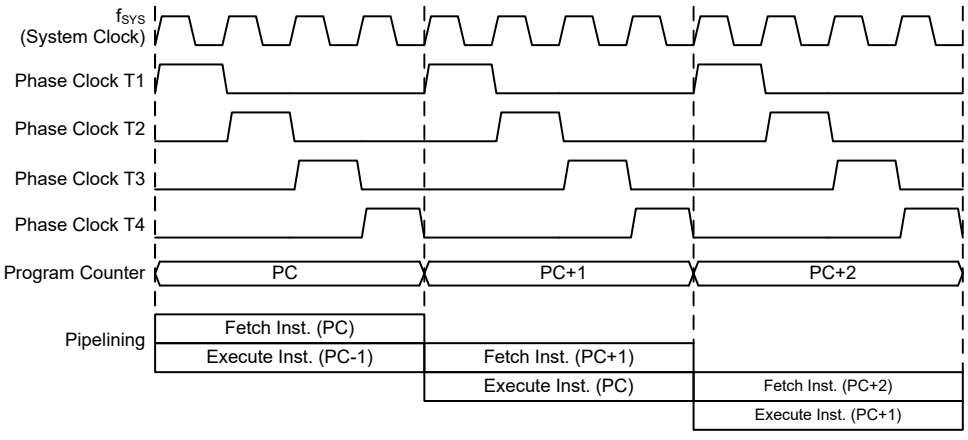
System Architecture

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 Periodic 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. 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.

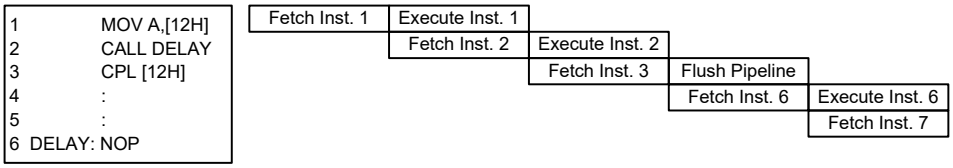
Clocking and Pipelining

The main system clock, derived from an 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.

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



System Clocking and Pipelining



Instruction Fetching

Program Counter

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

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

Program Counter	
High Byte	**Low Byte (PCL)**
PC9~PC8	PCL7~PCL0

Program Counter

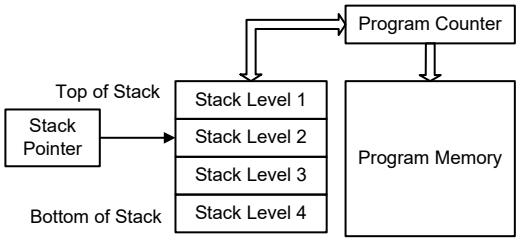
The lower byte of the Program Counter, known as the Program Counter Low register or PCL, is available for program control and is a readable and writeable register. By transferring data directly into this register, a short program jump can be executed directly; however, as only this low byte is available 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 4 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:

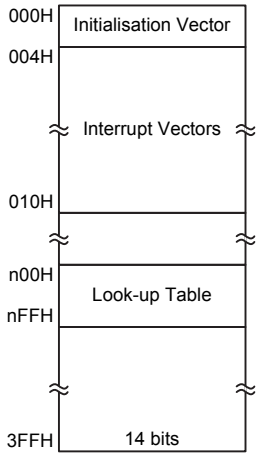
- Arithmetic operations:
  ADD, ADDM, ADC, ADCM, SUB, SUBM, SBC, SBCM, DAA
- Logic operations:
  AND, OR, XOR, ANDM, ORM, XORM, CPL, CPLA
- Rotation:
  RRA, RR, RRCA, RRC, RLA, RL, RLCA, RLC
- Increment and Decrement:
  INCA, INC, DECA, DEC
- Branch decision:
  JMP, SZ, SZA, SNZ, SIZ, SDZ, SIZA, SDZA,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

Flash Program Memory

The Program Memory is the location where the user code or program is stored. For the 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 offers 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 1K×14 bits. The Program Memory is addressed by the Program Counter and also contains data, table information and interrupt entries. Table data, which can be setup 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 bits in this transferred higher order byte will be read as “0”.

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

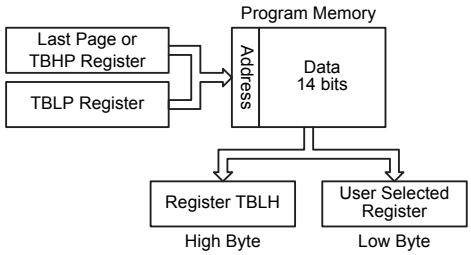


Table Program Example

The following example shows how the table pointer and table data is defined and retrieved from the 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 "300H" which refers to the start address of the last page within the 1K words Program Memory of the device. The table pointer 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 "306H" or 6 locations after the start of the last page. Note that the value for the table pointer is referenced to the first address of the present page 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          ; initialize low table pointer - note that this address is referenced
mov tblp,a         ; to the last page or present page
:
:
tabrdl tempreg1    ; transfers value in table referenced by table pointer data at program
                   ; memory address "306H" transferred to tempreg1
dec tblp           ; reduce value of table pointer by one
tabrdl tempreg2    ; transfers value in table referenced by table pointer data at program
                   ; memory address "305H" transferred to tempreg2 in this
                   ; example the data "1AH" is transferred to tempreg1 and data "0FH" to
                   ; register tempreg2
:
:
org 300h           ; sets initial address of program memory
db 0Ah, 0Bh, 0Ch, 0Dh, 0Eh, 0Fh, 1Ah, 1Bh
:
:
```

In Circuit Programming – ICP

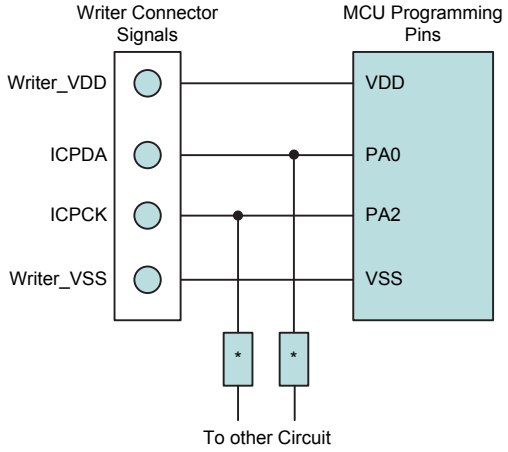
The provision of Flash type Program Memory provides the user with a means of convenient and easy upgrades and modifications to their programs on the same device.

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 is beyond the scope of this document and will be supplied in supplementary literature.

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



Note: * 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 HT45V5Q-1 which is used to emulate the real MCU device named HT45F5Q-1. The EV chip device also provides the "On-Chip Debug" function to debug the real MCU device during development process. The EV chip and real MCU device, HT45V5Q-1 and HT45F5Q-1, are almost functional compatible except the "On-Chip Debug" function. Users can use the EV chip device to emulate the real MCU device behaviors 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 device for debugging, the corresponding pin functions shared with the OCDSDA

and OCDSCK pins in the real MCU device will have no effect in the EV chip. However, the two OCDS pins which are pin-shared with the ICP programming pins are still used as the Flash Memory programming pins for ICP. 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 OCDS 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.

Divided into two types, the first of these is an area of RAM, known as the Special Function Data Memory. Here are located registers which 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.

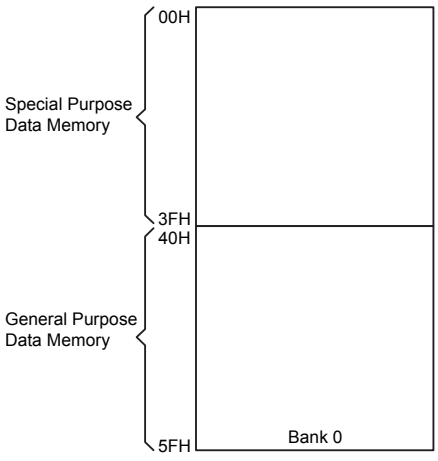
Structure

The Data Memory has only a bank named Bank 0, which is implemented in 8-bit wide Memory. The Data Memory Bank is categorized into two types, the special Purpose Data Memory and the General Purpose Data Memory.

The address range of the Special Purpose Data Memory for the device is from 00H to 3FH while the address range of the General Purpose Data Memory is from 40H to 5FH.

Special Purpose Data Memory		General Purpose Data Memory	
Located Bank	Bank: Address	Capacity	Bank: Address
0	0: 00H~3FH	32×8	0: 40H~5FH

Data Memory Summary



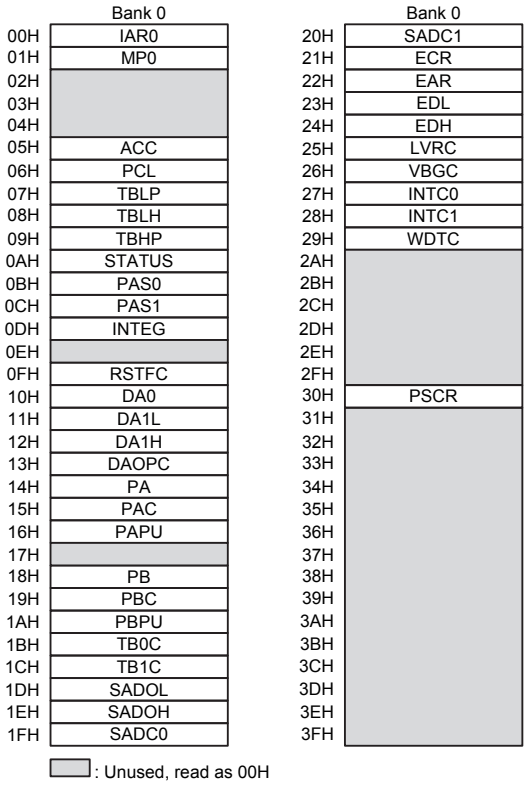
Data Memory Structure

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”.



Special Purpose Data Memory Structure

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 Register – IAR0

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

Memory Pointer – MP0

One Memory Pointer, known as MP0 is provided. The Memory Pointer is physically implemented in the Data Memory and can be manipulated in the same way as FAST registers providing a convenient way with which to address and track data. When any operation to the relevant Indirect Addressing Register 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.

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              ; setup size of block
     mov block, a
     mov a, offset adres1   ; Accumulator loaded with first RAM address
     mov mp0, a             ; setup memory pointer with first RAM address
loop:
     clr IAR0               ; clear the data at address defined by MP0
     inc mp0                ; increase memory pointer
     sdz block              ; check if last memory location has been cleared
     jmp loop
continue:
```

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

Accumulator – ACC

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

Program Counter Low 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 setup before any table read commands are executed. Their value can be changed, for example using the "INC" or "DEC" instruction, 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), 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 flag, 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 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" instruction.

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.
- **TO** is cleared by a system power-up or executing the "CLR WDT" 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	—	OV	Z	AC	C
R/W	—	—	R	—	R/W	R/W	R/W	R/W
POR	—	—	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" instruction
1: A watchdog time-out occurred.

Bit 4 Unimplemented, read as "0"

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 32×14 bits for the device. The Emulated EEPROM Erase operation is carried out in a page format while the Write and Read operations are carried out 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	1 word/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

Emulated EEPROM Registers

Four registers control the overall operation of the Emulated EEPROM Data Memory. These are the address register, EAR, the data registers, EDL and EDH, and a single control register, ECR.

Register Name	Bit							
	7	6	5	4	3	2	1	0
EAR	—	—	—	EAR4	EAR3	EAR2	EAR1	EAR0
EDL	D7	D6	D5	D4	D3	D2	D1	D0
EDH	—	—	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

• **EDL 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**: Emulated EEPROM data bit 7 ~ bit 0

• **EDH Register**

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

Bit 7~6 Unimplemented, read as "0"

Bit 5~0 **D13~D8**: Emulated EEPROM data bit 13 ~ 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 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 page erase operation is 16 words per page, 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 and desired write address should first be placed in the EDH/EDL and EAR registers respectively. 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 EDH/EDL 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.

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_DATA_L     ; user defined data
MOV  EDL, A
MOV  A, EEPROM_DATA_H
MOV  EDH, 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 set 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
MOV  A, EDL               ; are required move read data to register
MOV  READ_DATA_L, A
MOV  A, EDH
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.

Oscillators

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

Oscillator Overview

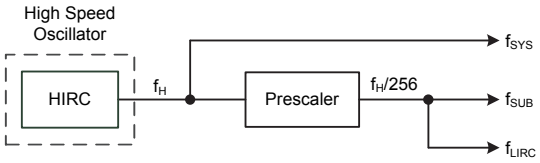
In addition to being the source of the main system clock the oscillator also provide clock sources for the Watchdog Timer and Time Base Interrupts. A fully integrated internal oscillator, requiring no external components, is provided to form system oscillator.

Type	Name	Frequency
Internal High Speed RC	HIRC	8MHz

Oscillator Type

System Clock Configurations

The device has a high speed oscillator to generate the system clock. The high speed oscillator is the internal 8MHz RC oscillator, known as the HIRC.



System Clock Configurations

Internal High Speed RC Oscillator – HIRC

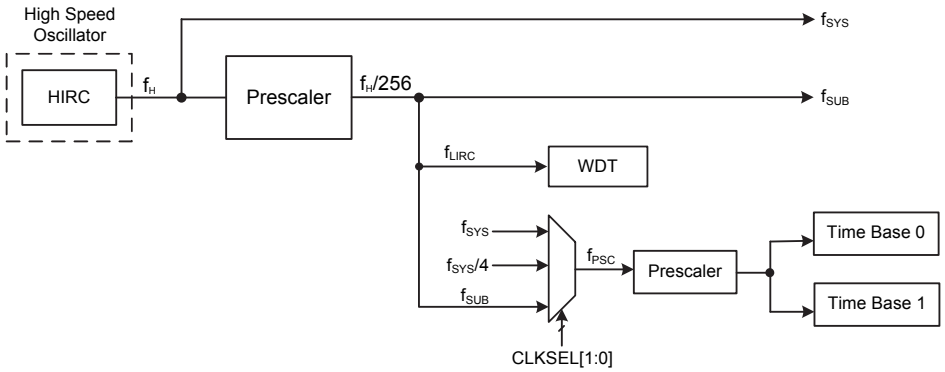
The internal RC oscillator is a fully integrated system oscillator requiring no external components. The internal RC oscillator has a fixed frequency of 8MHz. Device trimming during the manufacturing process and the inclusion of internal frequency compensation circuits are used to ensure that the influence of the power supply voltage, temperature and process variations on the oscillation frequency are also 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 clock required for high performance will by its nature increase current consumption

System Clocks

The system clock is come from a high frequency, f_H. The high speed system clock is sourced from the HIRC oscillator. A clock system can be configured to obtain maximum application performance.



Device Clock Configurations

System Operation Modes

There is one mode allowing FAST operation of the microcontroller, the FAST Mode.

FAST Mode

This mode operates allowing the microcontroller to operate FASTly with a clock source will come from the HIRC oscillator.

Watchdog Timer

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

Watchdog Timer Clock Source

The Watchdog Timer clock source is provided by the internal clock, f_{LIRC} which is sourced from the HIRC oscillator with the output frequency of $f_H/256$. The HIRC internal oscillator has a fixed frequency of 8MHz and this specified internal clock period can vary with V_{DD}, temperature and process variations. The Watchdog Timer source clock is then subdivided by a ratio of 2^8 to 2^{18} 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 timeout period as well as the enable and reset MCU operations. The WRF software reset flag will be indicated in the RSTFC register. These registers control the overall operation of the Watchdog Timer.

• 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	0	1	1

Bit 7~3 **WE4~WE0**: WDT function software control
10101 or 01010: Enable
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: $2^8/f_{LIRC}$
001: $2^{10}/f_{LIRC}$
010: $2^{12}/f_{LIRC}$
011: $2^{14}/f_{LIRC}$
100: $2^{15}/f_{LIRC}$
101: $2^{16}/f_{LIRC}$
110: $2^{17}/f_{LIRC}$
111: $2^{18}/f_{LIRC}$
These three bits determine the division ratio of the Watchdog Timer source clock, which in turn determines the timeout period. The internal clock, f_{LIRC} is supplied by the HIRC oscillator with the output frequency of $f_H/256$.

• 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
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 FAST 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 instructions. If the program malfunctions for whatever reason, jumps to an unknown location, or enters an endless loop, these clear instructions 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 control and reset control of the Watchdog Timer. The WDT function will be enabled if the WE4~WE0 bits are equal to 01010B or 10101B. 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.

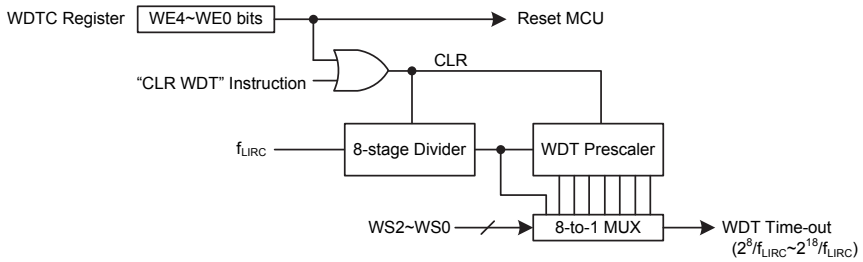
WE4~WE0 Bits	WDT Function
01010B or 10101B	Enable
Any other values	Reset MCU

Watchdog Timer Enable/Reset Control

Under FAST program operation, a Watchdog Timer time-out will initialise a device reset and set the status bit TO. Two 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.

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^{18} division ratio is selected. As an example, the Watchdog Timer clock source is provided by the internal clock, f_{LIRC}, which is supplied by the HIRC oscillator with the output frequency of $f_H/256$, this will give a maximum watchdog period of around 8 seconds for the 2^{18} 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.

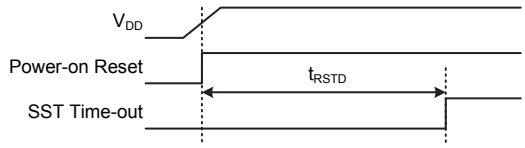
Another type of reset is when the Watchdog Timer overflows and resets the microcontroller. All types of reset operations result in different register conditions being setup.

Reset Functions

There are several ways in which a microcontroller reset can occur, through events occurring internally:

Power-on Reset

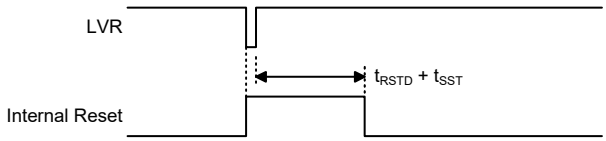
The most fundamental and unavoidable reset is the one that occurs after power is first applied to the microcontroller. As well as ensuring that the Program Memory begins execution from the first memory address, a power-on reset also ensures that certain other registers are preset to known conditions. 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. This function can be enabled or disabled by the LVRC control register. If the LVRC control register is configured to enable the LVR, the LVR function will be always enabled with a specific LVR voltage V_{LVR}. 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. If the LVS7~LVS0 bits are set to 01011010B, the LVR function is enabled with a fixed LVR voltage of 2.1V. If the LVS7~LVS0 bits are set to 10100101B, the LVR function is disabled. 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 01011010B.



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	1	0	1	0

Bit 7~3 Unimplemented, read as "0"

Bit 7~0 **LVS7~LVS0**: LVR voltage select control
01011010: 2.1V
10100101: Disable
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 01011010B and 10100101B, 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

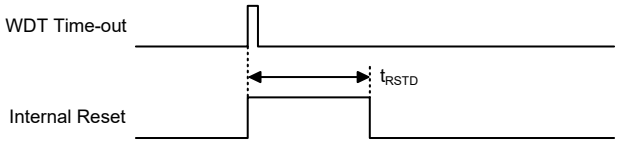
This bit is set high if the LVRC register contains any non-defined LVRC register values. This in effect acts like a software-reset function. This bit can only be cleared to zero by the application program.

Bit 0 **WRF**: WDT control register software reset flag

Refer to the Watchdog Timer Control Register section.

Watchdog Time-out Reset during FAST Mode

The Watchdog time-out Reset during normal operation in the FAST mode, the Watchdog time-out flag TO will be set to "1".



WDT Time-out Reset during FAST Mode Timing Chart

Reset Initial Conditions

The different types of reset described affect the reset flags in different ways. TO flag is located in the status register and is controlled by various microcontroller operations, such as the Watchdog Timer. The reset flag is shown in the table:

TO	RESET Conditions
0	Power-on reset
u	LVR reset during FAST Mode operation
1	WDT time-out reset during FAST Mode operation

"u": unchanged

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

Item	Condition after Reset
Program Counter	Reset to zero
Interrupts	All interrupts will be disabled
WDT, Time Bases	Clear after reset, WDT begins counting
Input/Output Ports	I/O ports will be setup 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 FAST 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	WDT Time-out
IAR0	xxxx xxxx	uuuu uuuu
MP0	xxxx xxxx	uuuu uuuu
ACC	xxxx xxxx	uuuu uuuu
PCL	0000 0000	0000 0000
TBLP	xxxx xxxx	uuuu uuuu
TBLH	--xx xxxx	--uu uuuu
TBHP	---- --xx	---- --uu
STATUS	--0- xxxx	--1- uuuu
PAS0	0000 0000	0000 0000
PAS1	0000 --00	0000 --00
INTEG	---- --00	---- --00
RSTFC	---- -x00	---- -uuu
DA0	0000 0100	0000 0100
DA1L	0000 0000	0000 0000
DA1H	---- 1000	---- 1000

Register	Power on Reset	WDT Time-out
DAOPC	11-- ----	11-- ----
PA	1111 1111	1111 1111
PAC	1111 1111	1111 1111
PAPU	0000 0000	0000 0000
PB	---- ---1	---- ---1
PBC	---- ---1	---- ---1
PBPU	---- ---0	---- ---0
TB0C	0--- -000	0--- -000
TB1C	0--- -000	0--- -000
SADOL	xx-- ----	xx-- ----
SADOH	xxxx xxxx	xxxx xxxx
SADC0	0000 0000	0000 0000
SADC1	0000 0000	00000000
ECR	0000 0000	0000 0000
EAR	---0 0000	---0 0000
EDL	0000 0000	0000 0000
EDH	--00 0000	--00 0000
LVRC	0101 1010	0101 1010
VBGC	---- 0---	---- 0---
INTC0	-000 0000	-000 0000
INTC1	--0 ---0	---0 ---0
WDTC	0101 0011	0101 0011
PSCR	---- --00	---- --00

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, the user is provided with an I/O structure to meet the needs of a wide range of application possibilities.

The device provides bidirectional input/output lines labeled with port names PA and PB. 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
PB	—	—	—	—	—	—	—	PB0
PBC	—	—	—	—	—	—	—	PBC0
PBPU	—	—	—	—	—	—	—	PBPU0

“—”: 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 input have the capability of being connected to an internal pull-high resistor. These pull-high resistors are selected using registers, namely PAPU and PBPU, 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 an 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 or B. However, the actual available bits for each I/O Port may be different.

I/O Port Control Registers

Each I/O port has its own control register known as PAC and PBC, 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 setup as a CMOS output. If the pin is currently setup 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 or B. However, the actual available bits for each I/O Port may be different.

Pin-shared Functions

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

Pin-shared Function Selection Registers

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

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

Register Name	Bit							
	7	6	5	4	3	2	1	0
PAS0	PAS07	PAS06	PAS05	PAS04	PAS03	PAS02	PAS01	PAS00
PAS1	PAS17	PAS16	PAS15	PAS14	—	—	PAS11	PAS10

Pin-shared Function Selection Register List

- **PAS0 Register**

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

Bit 7~6 **PAS07~PAS06**: PA3 Pin-Shared function selection
00/01: PA3
10: VREF
11: AN2

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

Bit 3~2 **PAS03~PAS02**: PA1 Pin-Shared function selection
00/01/10: PA1
11: AN4

Bit 1~0 **PAS01~PAS00**: PA0 Pin-Shared function selection
00/01/10: PA0
11: AN0

• **PAS1 Register**

Bit	7	6	5	4	3	2	1	0
Name	PAS17	PAS16	PAS15	PAS14	—	—	PAS11	PAS10
R/W	R/W	R/W	R/W	R/W	—	—	R/W	R/W
POR	0	0	0	0	—	—	0	0

Bit 7~6 **PAS17~PAS16**: PA7 Pin-Shared function selection
00/01/10: PA7
11: OPA1P

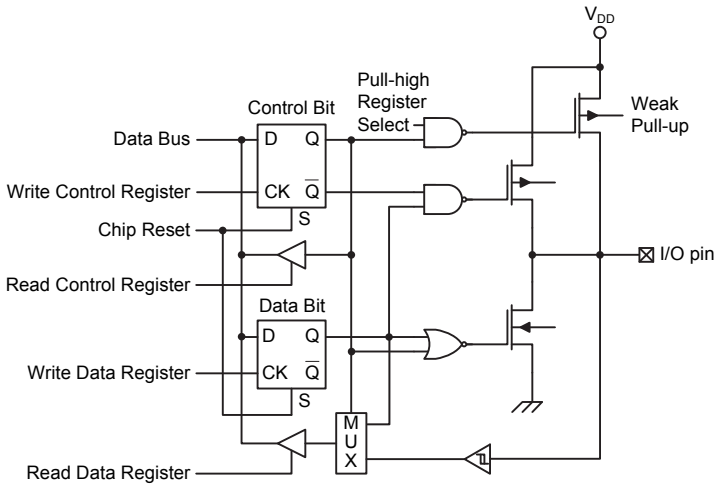
Bit 5~4 **PAS15~PAS14**: PA6 Pin-Shared function selection
00/01/10: PA6
11: OPA0P

Bit 3~2 Unimplemented, read as "0"

Bit 1~0 **PAS11~PAS10**: PA4 Pin-Shared function selection
00/01/10: PA4
11: AN3

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 setup 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.

Analog to Digital Converter

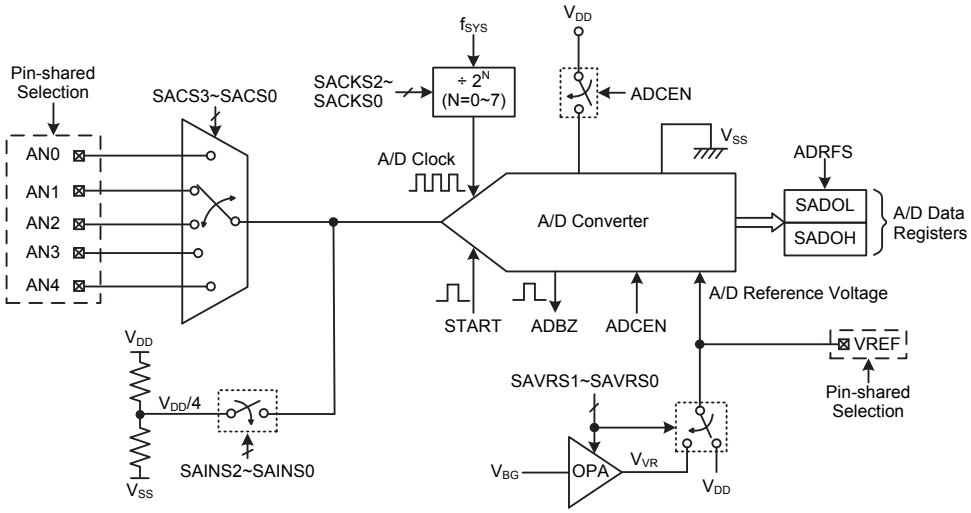
The need to interface to real world analog signals is a common requirement for many electronic systems. However, to properly process these signals by a microcontroller, they must first be converted into digital signals by A/D converters. By integrating the A/D conversion electronic circuitry into the 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

This 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, or the internal analog signal, the A/D power divided by 4, $V_{DD}/4$, and convert these signals directly into a 10-bit digital value. The external analog signal to be converted is determined by the SACS3~SACS0 and SAINS2~SAINS0 bits. When the external analog signal is to be converted, the corresponding pin-shared control bits should first be properly configured and then desired external channel input should be selected using the SACS3~SACS0 and SAINS2~SAINS0 bits. More detailed information about the A/D input signal is described in the "A/D Converter Control Registers" and "A/D Converter Input Signals" sections respectively.

External Input Channels	Internal Signal	Channel Select Bits
5: AN0~AN4	$V_{DD}/4$	SAINS2~SAINS0 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 several 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. An additional register VBGC is used to control the Bandgap reference voltage on/off.

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	SAINS2	SAINS1	SAINS0	SAVRS1	SAVRS0	SACKS2	SACKS1	SACKS0
VBGC	—	—	—	—	VBGEN	—	—	—

A/D Converter Register List

A/D Converter Data Registers – SADOL, SADOH

As this device contains an internal 10-bit A/D converter, 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 A/D 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 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 SAINS2~SAINS0 bits in the SADC1 register are used to determine that the analog signal to be converted comes from the internal analog signal or external analog channel input. 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 A/D conversion
This bit is used to initiate an A/D conversion process. The bit is FASTly low but if set high and then cleared low again, the A/D converter will initiate a conversion process.

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

Bit 5 **ADCEN**: A/D converter function enable control
0: Disable
1: Enable
This bit controls the A/D internal function. This bit should be set to one to enable the A/D converter. If the bit is 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 data format select
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 select
0000: AN0
0001: AN1
0010: AN2
0011: AN3
0100: AN4
0101~1111: Non-existed channel, the input will be floating if selected

• **SADC1 Register**

Bit	7	6	5	4	3	2	1	0
Name	SAINS2	SAINS1	SAINS0	SAVRS1	SAVRS0	SACKS2	SACKS1	SACKS0
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 7~5 **SAINS2~SAINS0**: A/D converter input signal selection
000: External input – External analog channel input
001~011: Unused, connected to ground
100: Internal input – Internal A/D power supply voltage divided by 4, $V_{DD}/4$
101~111: External input – External analog channel input
Care must be taken if the SAINS2~SAINS0 bits are set to "100" to select the internal analog signal to be coverted. When the internal analog signal is selected to be converted, the external input pin must never be selected as the A/D input signal by properly setting the SACS3~SACS0 bits with a value from 0101 to 1111. Otherwise, the external channel input will be connected together with the internal analog signal. This will result in unpredictable situations such as an irreversible damage.

Bit 4~3 **SAVRS1~SAVRS0**: A/D converter reference voltage select
00: VREF pin
01: Internal A/D converter power, V_{DD}
10: Internal OPA output, V_{VR}
11: External VREF pin
These bits are used to select the A/D converter reference voltage. Care must be taken if the SAVRS1~SAVRS0 bits are set to "01" or "10" to select the A/D converter power supply or OPA output as the reference voltage source. When the internal A/D converter power is selected as the reference voltage, the VREF pin cannot be configured as the reference voltage input by properly configuring the corresponding pin-shared function control bits. Otherwise, the external input voltage on VREF pin will be connected to the internal A/D converter power.

Bit 2~0 **SACKS2~SACKS0**: A/D conversion clock source select
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: $f_{SYS}/128$
These three bits are used to select the clock source for the A/D converter.

• VBGC Register

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

Bit 7~4 Unimplemented, read as "0"

Bit 3 **VBGEN**: V_{BG} Bandgap reference control
0: Disable
1: Enable
Note that the Bandgap circuit is enabled when the LVR function is enabled or when the VBGEN bit is set high.

Bit 2~0 Unimplemented, read as "0"

A/D Converter Operation

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

The ADBZ bit in the SADC0 register is used to indicate whether the analog to digital conversion process is in progress or not. This bit will be automatically set to 1 by the microcontroller after an A/D conversion is successfully initiated. When the A/D conversion is complete, the ADBZ 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 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, as the system clock operates at a frequency of 8MHz, the SACKS2~SACKS0 bits should not be set to 000, 001 or 111. Doing so will give A/D clock periods that are less than the minimum A/D clock period or greater than the maximum A/D clock period which may result in inaccurate A/D conversion values. Refer to the following table for examples, where values marked with an asterisk * show where, depending upon the device, special care must be taken, as the values may be exceeding the specified minimum A/D Clock Period range.

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

A/D Clock Period Examples

Controlling the power on/off function of the A/D converter circuitry is implemented using the ADCEN bit in the SADC0 register. This bit must be set high to power on the A/D converter. When the ADCEN bit is set high to power on the A/D converter internal circuitry a certain delay, as indicated in the timing diagram, must be allowed before an A/D conversion is initiated. Even if no pins are selected for use as A/D inputs, 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.

A/D Converter Reference Voltage

The reference voltage supply to the A/D converter can be supplied from the internal A/D power supply voltage, V_{DD}, or internal operational amplifer output voltage, V_{VR}, or from an external reference source supplied on pin VREF. The desired selection is made using the SAVRS1~SAVRS0 bits. When the SAVRS bit field is set to “01”, the A/D converter reference voltage will come from the power supply voltage. When the SAVRS1~SAVRS0 bits are set to “10”, the A/D converter reference voltage will come from the internal operational amplifer output voltage, V_{VR}. Otherwise, if the SAVRS1~SAVRS0 bits are set to other value except “01” and “10”, the A/D converter reference voltage will come from the VREF pin. As the VREF pin is pin-shared with other functions, when the VREF pin is selected as the reference voltage supply pin, the VREF pin-shared function control bit should be properly configured to disable other pin functions. However, if the A/D power supply or the operational amplifier output voltage is selected as the reference voltage, the VREF pin must not be configured as the reference voltage input function to avoid the internal connection between the VREF pin and the internal reference signal. The analog input values must not be allowed to exceed the selected reference voltage.

SAVRS[1:0]	Reference	Description
00/11	VREF pin	External A/D converter reference pin VREF
01	V_{DD}	Internal A/D converter power supply voltage
10	V_{VR}	Internal operational amplifier output voltage

A/D Converter Reference Voltage Selection

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 PAS0 and PAS1 register determine whether the input pins are setup as A/D converter analog inputs or whether they have other functions. If the pin is setup 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 setup through register programming, will be automatically disconnected if the pins are setup as A/D inputs. Note that it is not necessary to first setup 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.

If the SAINS2~SAINS0 bits are set to "000" or "101~111", the external analog channel input is selected to be converted and the SACS3~SACS0 bits can determine which actual external channel is selected to be converted. If the SAINS2~SAINS0 bits are set to "100", the $V_{DD}/4$ voltage is selected to be converted. Note that if the internal analog signal is selected to be converted, the external input channel determined by the SACS3~SACS0 bits must be switched to a non-existed A/D input channel by properly setting the SACS3~SACS0 bits with a value from "0101" to "1111".

SACS[3:0]	SACS[3:0]	Input Signals	Description
000, 101~111	0000~0100	AN0~AN4	External channel analog input
	0101~1111	–	Floating, no external channel is selected
001~011	0100~1111	–	Unused, connected to ground
100	0100~1111	$V_{DD}/4$	Internal A/D converter power supply voltage/4, $V_{DD}/4$

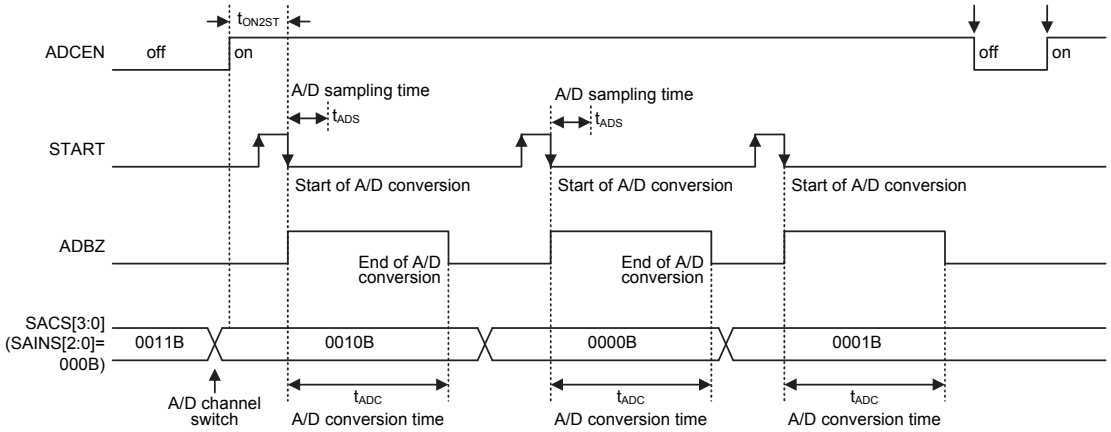
A/D Converter Input Signal Selection

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.

Maximum single A/D conversion rate = A/D clock period/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 – External Channel Input

Summary of A/D Conversion Steps

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

- Step 1
  Select the required A/D conversion clock by correctly programming bits SACKS2~SACKS0 in the SADC1 register.
- Step 2
  Enable the A/D by setting the ADCEN bit in the SADC0 register to one.
- Step 3
  Select which signal is to be connected to the internal A/D converter by correctly configuring the SAINS2~SAINS0 bits in the SADC1 register.
  Select the external channel input to be converted, go to Step 4.
  Select the internal analog signal to be converted, go to Step 5.
- Step 4
  If the A/D input signal comes from the external channel input selected by configuring the SAINS2~SAINS0 bits, the corresponding pin should be configured as A/D input function by configuring the relevant pin-shared function control bits. The desired analog channel then should be selected by configuring the SACS3~SACS0 bits. After this step, go to Step 6.
- Step 5
  Before the A/D input signal is selected to come from the internal analog signal by configuring the SAINS2~SAINS0 bits, the corresponding external input pin must be switched to a non-existed channel input by setting the SACS3~SACS0 bits with a value from 0101 to 1111. The desired internal analog signal then can be selected by configuring the SAINS2~SAINS0 bits. After this step, go to Step 6.
- Step 6
  Select the reference voltage source by configuring the SAVRS1~SAVRS0 bits in the SADC1 register. If the A/D power supply voltage or the operational amplifier output voltage is selected, the external reference input pin function must be disabled by properly configuring the corresponding pin-shared control bits.
- Step 7
  Select A/D converter output data format by setting the ADRFS bit in the SADC0 register.

- Step 8
  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 9
  The A/D conversion procedure can now be initialized by setting the START bit from low to high and then low again.
- Step 10
  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 FAST 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

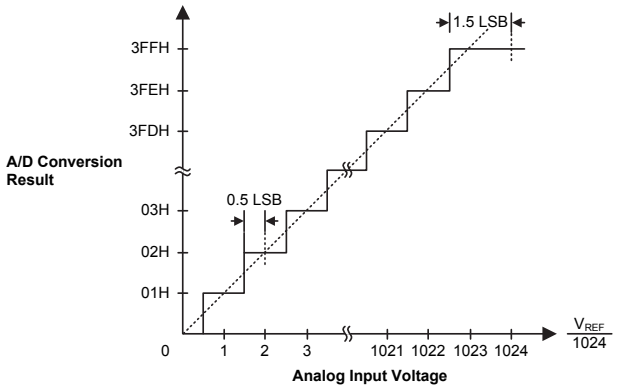
As the device contains a 10-bit A/D converter, its full-scale converted digitised value is equal to 3FFH. Since the full-scale analog input value is equal to the actual A/D converter reference voltage, V_{REF}, this gives a single bit analog input value of V_{REF} divided by 1024.

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

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 determined by the SAVRS field.



Ideal A/D Transfer Function

A/D Conversion Programming Examples

The following two programming examples illustrate how to setup 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 and
                        ; select external channel input and external reference input
mov  SADC1,a
set  ADCEN
mov  a,03h              ; set PAS0 to configure pin AN0 and pin VREF
mov  PAS0,a
mov  a,20h
mov  SADC0,a            ; enable 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
```

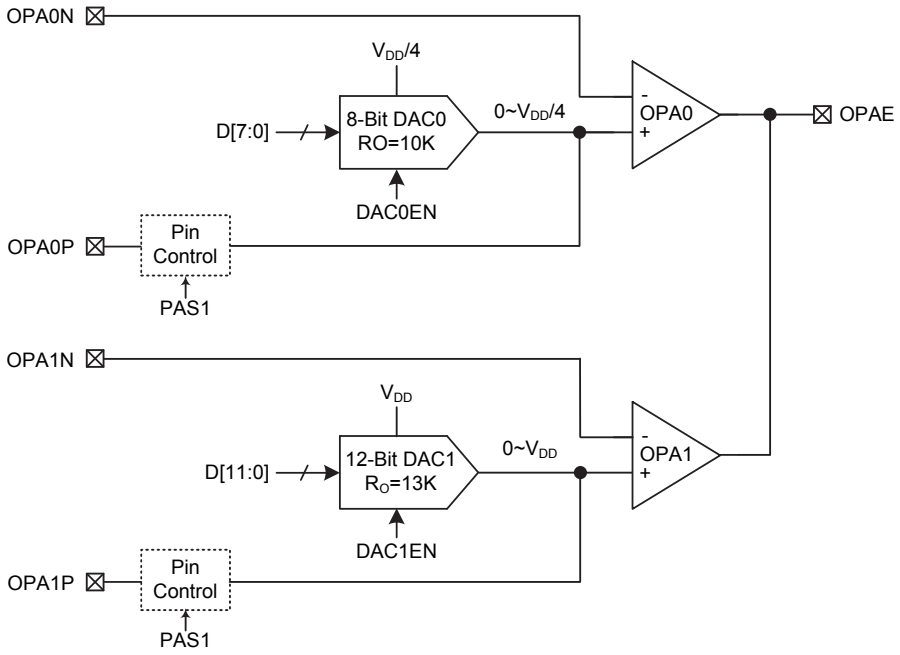
Example: using the interrupt method to detect the end of conversion

```
clr  ADE                ; disable ADC interrupt
mov  a,03H              ; select fSYS/8 as A/D clock and
                        ; select external channel input and external reference input
mov  SADC1,a
set  ADCEN
mov  a,03h              ; set PAS0 to configure pin AN0 and pin VREF
mov  PAS0,a
mov  a,20h
mov  SADC0,a            ; enable and connect AN0 channel to A/D converter
Start_conversion:
clr  START              ; high pulse on START bit to initiate conversion
set  START              ; reset A/D
clr  START              ; start A/D
clr  ADF                ; clear ADC interrupt request flag
set  ADE                ; enable ADC interrupt
set  EMI                ; enable global interrupt
:
:
; ADC interrupt service routine
ADC_ISR:
mov  acc_stack,a        ; save ACC to user defined memory
mov  a,STATUS
mov  status_stack,a     ; save STATUS to user defined memory
:
:
mov  a,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
```

Battery Charge Module

The device contains a battery charge module which consists of two operational amplifiers, an 8-bit D/A converter (DAC0) and a 12-bit D/A converter (DAC1). DAC0 and DAC1 converters are used to set constant current (CC) and constant voltage (CV) respectively. In addition, the operational amplifiers are always enabled.

The structure of the battery charge module is shown as below.



Note: 1. The operational amplifiers with open drain outputs.
2. The operational amplifiers do not need calibration offset.

Battery Charge Module Block Diagram

Digital to Analog Converter

The battery charge module contains an 8-bit (DAC0) and a 12-bit (DAC1) D/A converters with R2R architectures. The D/A converters can be power down to save power. DAC0 output voltage range is from 0 to $V_{DD}/4$ while DAC1 output voltage range is from 0 to V_{DD}. DACnOUT is D/A converter analog output voltage. The following shows how the DACnOUT value can be calculated.

DAC0OUT = (DAC $V_{REF0}/2^8$) × DA0[7:0] where $V_{REF0} = V_{DD}/4$;

DAC1OUT = (DAC $V_{REF1}/2^{12}$) × DA1[11:0] where $V_{REF1} = V_{DD}$.

Battery Charge Module Registers

The overall operation of the battery charge module is controlled by several registers and the corresponding register definitions are described in the accompanying sections.

Register Name	Bit							
	7	6	5	4	3	2	1	0
DA0	D7	D6	D5	D4	D3	D2	D1	D0
DA1L	D7	D6	D5	D4	D3	D2	D1	D0
DA1H	—	—	—	—	D11	D10	D9	D8
DAOPC	DAC1EN	DAC0EN	—	—	—	—	—	—

• **DA0 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	1	0	0

Bit 7~0 **D7~D0**: D/A converter 0 output control code
The D/A converter output is calculated using the following equation:
DAC0OUT = (DAC $V_{REF0}/2^8$) × D[7:0] where $V_{REF0} = V_{DD}/4$.

• **DA1L & DA1H Registers**

Register	DA1H								DA1L							
Bit	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
Name	—	—	—	—	D11	D10	D9	D8	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	R/W	R/W	R/W	R/W
POR	—	—	—	—	1	0	0	0	0	0	0	0	0	0	0	0

"—": Unimplemented, read as "0"

D11~D8: D/A converter 1 output control code high byte

D7~D0: D/A converter 1 output control code low byte

The D/A converter output voltage is calculated using the following equation:

DAC1OUT = (DAC $V_{REF1}/2^{12}$) × D[11:0] where $V_{REF1} = V_{DD}$.

Write the DA1L register only write to shadow buffer, and until write DA1H register will also copy the shadow buffer data to DA1L register.

• **DAOPC Register**

Bit	7	6	5	4	3	2	1	0
Name	DAC1EN	DAC0EN	—	—	—	—	—	—
R/W	R/W	R/W	—	—	—	—	—	—
POR	1	1	—	—	—	—	—	—

Bit 7 **DAC1EN**: D/A Converter 1 Enable Control Bit
0: Disable
1: Enable

Bit 6 **DAC0EN**: D/A Converter 0 Enable Control Bit
0: Disable
1: Enable

Bit 5~0 Unimplemented, read as "0"

Note: When D/A converter is disabled, the output will be floating.

Operational Amplifiers

The battery charge module contains two operational amplifiers, namely OPA0 and OPA1. The operational amplifiers are always enabled. OPA0 and OPA1 are specifically for constant current (CC) and constant voltage (CV) control respectively.

Interrupts

Interrupts are an important part of any microcontroller system. When an external event or an internal function such as 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 an external interrupt and several internal interrupt functions. The external interrupt is generated by the action of the external INT pin, while the internal interrupts are generated by various internal functions such as the Time Bases 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 number of registers falls into two categories. The first is the INTC0~INTC1 registers which setup the primary interrupts. The second 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	—
A/D Converter	ADE	ADF	—
Time Base	TBnE	TBnF	n=0~1

Interrupt Register Bit Naming Conventions

Register Name	Bit							
	7	6	5	4	3	2	1	0
INTEG	—	—	—	—	—	—	INTS1	INTS0
INTC0	—	TB1F	TB0F	INTF	TB1E	TB0E	INTE	EMI
INTC1	—	—	—	ADF	—	—	—	ADE

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	—	TB1F	TB0F	INTF	TB1E	TB0E	INTE	EMI
R/W	—	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	—	0	0	0	0	0	0	0

Bit 7 Unimplemented, read as "0"

Bit 6 **TB1F**: Time Base 1 interrupt request flag
0: No request
1: Interrupt request

Bit 5 **TB0F**: Time Base 0 interrupt request flag
0: No request
1: Interrupt request

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

Bit 3 **TB1E**: Time Base 1 interrupt control
0: Disable
1: Enable

Bit 2 **TB0E**: Time Base 0 interrupt control
0: Disable
1: Enable

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	—	—	—	ADE
R/W	—	—	—	R/W	—	—	—	R/W
POR	—	—	—	0	—	—	—	0

Bit 7~5 Unimplemented, read as "0"

Bit 4 **ADF**: A/D Converter interrupt request flag
0: No request
1: Interrupt request

Bit 3~1 Unimplemented, read as "0"

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

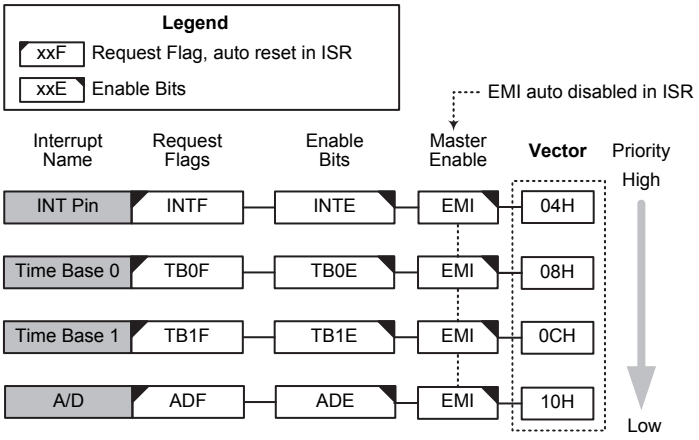
Interrupt Operation

When the conditions for an interrupt event occur, such as an A/D conversion completion 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 FAST 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. These interrupt sources have their own individual 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.



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 the 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. The pin must also be setup 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 the 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.

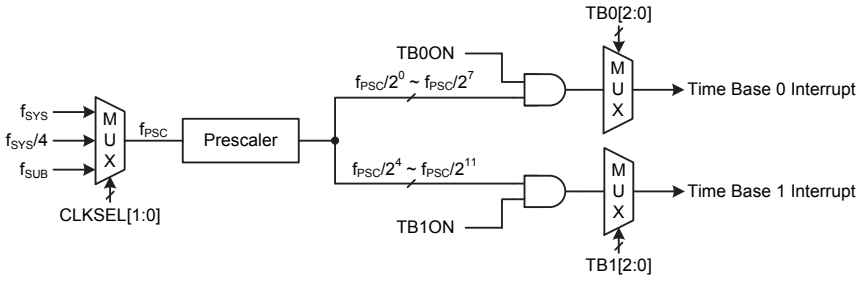
A/D Converter Interrupt

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

Time Base Interrupts

The function of the Time Base Interrupts is to provide regular time signal in the form of an internal interrupt. They are controlled by the overflow signals from their respective timer functions. When these happens their respective interrupt request flags, TB0F or TB1F will be set. To allow the program to branch to their respective interrupt vector addresses, the global interrupt enable bit, EMI and Time Base enable bits, TB0E or TB1E, must first be set. When the interrupt is enabled, the stack is not full and the Time Base overflows, a subroutine call to their respective vector locations will take place. When the interrupt is serviced, the respective interrupt request flag, TB0F or TB1F, will be automatically reset and the EMI bit will be cleared to disable other interrupts.

The purpose of the Time Base Interrupt is to provide an interrupt signal at fixed time periods. Its clock source, f_{PSC}, originates from the internal clock source f_{SYS}, $f_{SYS}/4$ or f_{SUB} and then passes through a divider, the division ratio of which is selected by programming the appropriate bits in the TB0C and TB1C registers to obtain longer interrupt periods whose value ranges. The clock source which in turn controls the Time Base interrupt period is selected using the CLKSEL1~CLKSEL0 bits in the PSCR register. This internal clock f_{SUB} is sourced by the HIRC oscillator with the output frequency of $f_H/256$.



Time Base Interrupts

• **PSCR Register**

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

Bit 7~2 Unimplemented, read as "0"

Bit 1~0 **CLKSEL1~CLKSEL0**: Prescaler clock source selection
00: f_{SYS}
01: $f_{SYS}/4$
1x: f_{SUB}

• **TB0C Register**

Bit	7	6	5	4	3	2	1	0
Name	TB0ON	—	—	—	—	TB02	TB01	TB00
R/W	R/W	—	—	—	—	R/W	R/W	R/W
POR	0	—	—	—	—	0	0	0

Bit 7 **TB0ON**: Time Base 0 Control
0: Disable
1: Enable

Bit 6~3 Unimplemented, read as "0"

Bit 2~0 **TB02~TB00**: Select Time Base 0 Time-out Period
000: $2^0/f_{PSC}$
001: $2^1/f_{PSC}$
010: $2^2/f_{PSC}$
011: $2^3/f_{PSC}$
100: $2^4/f_{PSC}$
101: $2^5/f_{PSC}$
110: $2^6/f_{PSC}$
111: $2^7/f_{PSC}$

• **TB1C Register**

Bit	7	6	5	4	3	2	1	0
Name	TB1ON	—	—	—	—	TB12	TB11	TB10
R/W	R/W	—	—	—	—	R/W	R/W	R/W
POR	0	—	—	—	—	0	0	0

Bit 7 **TB1ON**: Time Base 1 Control
0: Disable
1: Enable

Bit 6~3 Unimplemented, read as "0"

Bit 2~0 **TB12~TB10**: Select Time Base 1 Time-out Period
000: $2^4/f_{PSC}$
001: $2^5/f_{PSC}$
010: $2^6/f_{PSC}$
011: $2^7/f_{PSC}$
100: $2^8/f_{PSC}$
101: $2^9/f_{PSC}$
110: $2^{10}/f_{PSC}$
111: $2^{11}/f_{PSC}$

Programming Considerations

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

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

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.

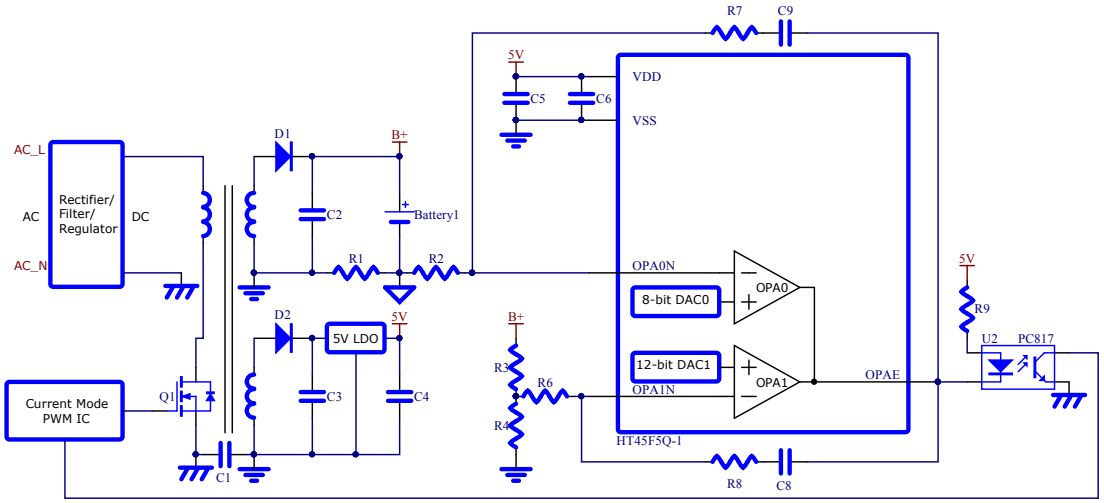
Application Descriptions

Introduction

According to the battery current condition, the charger can use a Buck circuit to implement charger management. The battery charging contains Constant Voltage Mode and Constant Current Mode. The device is an MCU specifically designed for battery charger applications. The abovementioned function control can be implemented by the integrated battery charger module, the details are described below.

Functional Description

The device contains a battery charge module which consists of two operational Amplifier functions, an 8-bit D/A Converter with an output voltage range from 0V to $V_{DD}/4$ (DAC0) and a 12-bit D/A Converter with an output voltage range from 0V to V_{DD} (DAC1) functions. The open drain OPA0~OPA1 and DAC0~DAC1 are used for constant voltage and constant current signals control. The OPA output can directly drive the photo-coupler, which makes the PWM IC on the primary side can implement output power adjustment, shown in the figure below.



Constant Voltage Mode Description

Constant voltage charging means that the charge voltage will remain at a constant value no matter how the battery internal resistance changes. The principle is that the charge voltage B+ is divided by R3 and R4 resistors and then supplied to the OPA1 negative terminal through the OPA1N. The difference between the OPA1N voltage and the D/A converter voltage is amplified and then output on the OPAE pin. This output will be sent to the PWM IC via a photo-coupler. If the OPA1N voltage is lower than the D/A Converter voltage (DAC1), the PWM IC will increase the PWM duty cycle and vice versa.

Note: The DAC1 (DA1H and DA1L) are used to set the maximum voltage threshold.

Constant Current Mode Description

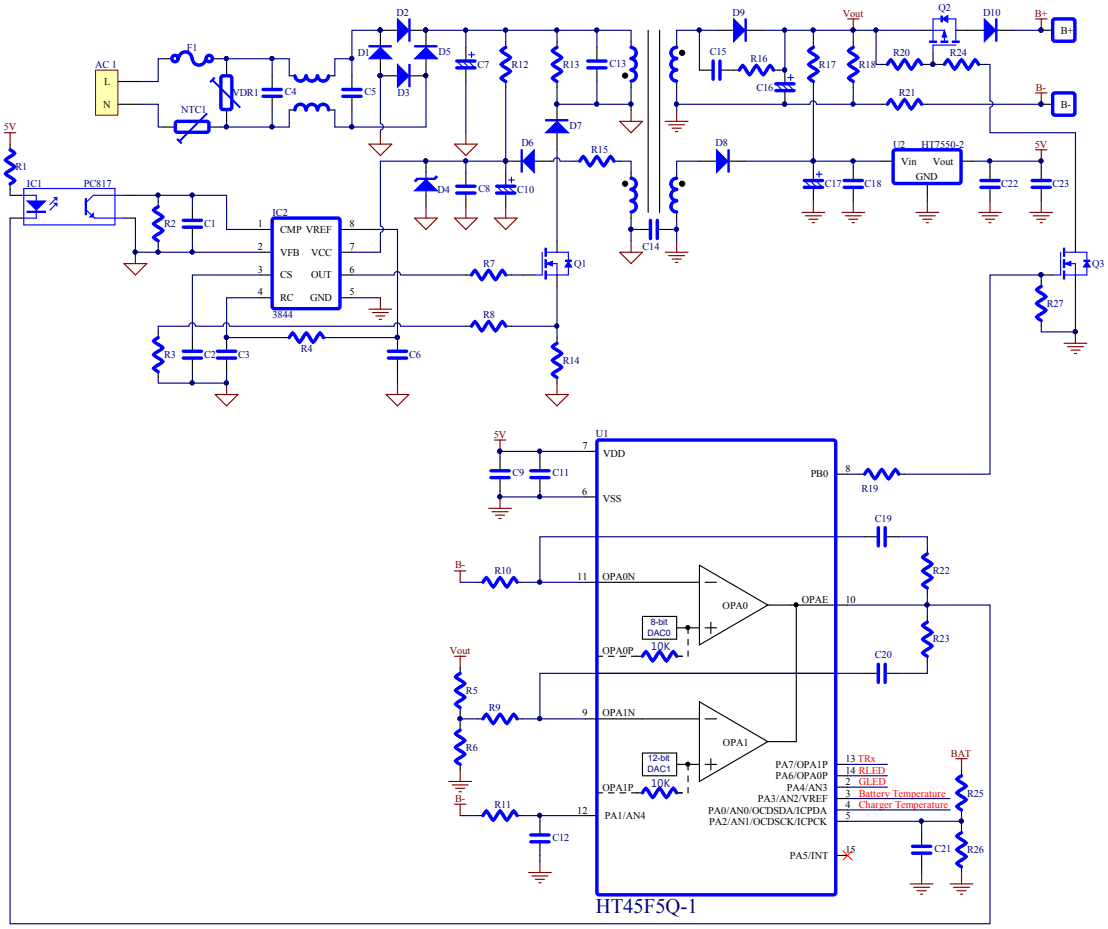
Constant current charging means that the charge current will remain at a constant value no matter how the battery internal resistance changes. The principle is that the charge current flows through the detecting resistor R1 and in turn generates a voltage, which will be input to the OPA0 negative terminal through the OPA0N. The difference between the OPA0N voltage and the D/A converter voltage is amplified and then output on the OPAE pin. This output will be sent to the PWM IC via a photo-coupler. If the OPA0N voltage is lower than the D/A Converter voltage (DA0), the PWM IC will increase the PWM duty cycle and vice versa.

Note: The DAC0 (DA0) is used to set the maximum current threshold.

Constant Voltage and Constant Current Resolution Increasing Method

If the internal 8/12-bit D/A Converter resolution is not high enough, the OPA0 and OPA1 positive terminals can be supplied by an external divider resistor to increase the voltage and current resolution.

Hardware Circuit



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

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

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

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

Instruction Definition

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

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

Operation ACC ← ACC + [m] + C

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

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

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

Operation [m] ← ACC + [m] + C

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

ADD A,[m] Add Data Memory to ACC

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

Operation ACC ← ACC + [m]

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

ADD A,x Add immediate data to ACC

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

Operation ACC ← ACC + x

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

ADDM A,[m] Add ACC to Data Memory

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

Operation [m] ← ACC + [m]

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

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

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

Operation ACC ← ACC "AND" [m]

Affected flag(s) Z

AND A,x Logical AND immediate data to ACC

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

Operation ACC ← ACC "AND" x

Affected flag(s) Z

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

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

Operation [m] ← ACC "AND" [m]

Affected flag(s) Z

CALL addr Subroutine call

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

Operation Stack ← Program Counter + 1
Program Counter ← addr

Affected flag(s) None

CLR [m] Clear Data Memory

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

Operation [m] ← 00H

Affected flag(s) None

CLR [m].i Clear bit of Data Memory

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

Operation [m].i ← 0

Affected flag(s) None

CLR WDT Clear Watchdog Timer

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

Operation WDT cleared
TO ← 0
PDF ← 0

Affected flag(s) TO, PDF

CPL [m] Complement Data Memory

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

Operation [m] ← $\overline{[m]}$

Affected flag(s) Z

CPLA [m] Complement Data Memory with result in ACC

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

Operation ACC ← $\overline{[m]}$

Affected flag(s) Z

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

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

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

Affected flag(s) C

DEC [m] Decrement Data Memory

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

Operation [m] ← [m] − 1

Affected flag(s) Z

DECA [m] Decrement Data Memory with result in ACC

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

Operation ACC ← [m] − 1

Affected flag(s) Z

HALT Enter power down mode

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

Operation TO ← 0
PDF ← 1

Affected flag(s) TO, PDF

INC [m] Increment Data Memory

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

Operation [m] ← [m] + 1

Affected flag(s) Z

INCA [m] Increment Data Memory with result in ACC

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

Operation ACC ← [m] + 1

Affected flag(s) Z

JMP addr Jump unconditionally

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

Operation Program Counter ← addr

Affected flag(s) None

MOV A,[m] Move Data Memory to ACC

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

Operation ACC ← [m]

Affected flag(s) None

MOV A,x Move immediate data to ACC

Description The immediate data specified is loaded into the Accumulator.

Operation ACC ← x

Affected flag(s) None

MOV [m],A Move ACC to Data Memory

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

Operation [m] ← ACC

Affected flag(s) None

NOP No operation

Description No operation is performed. Execution continues with the next instruction.

Operation No operation

Affected flag(s) None

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

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

Operation ACC ← ACC "OR" [m]

Affected flag(s) Z

OR A,x Logical OR immediate data to ACC

Description Data in the Accumulator and the specified immediate data perform a bitwise logical OR operation. The result is stored in the Accumulator.

Operation ACC ← ACC "OR" x

Affected flag(s) Z

ORM A,[m] Logical OR ACC to Data Memory

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

Operation [m] ← ACC "OR" [m]

Affected flag(s) Z

RET Return from subroutine

Description The Program Counter is restored from the stack. Program execution continues at the restored address.

Operation Program Counter ← Stack

Affected flag(s) None

RET A,x Return from subroutine and load immediate data to ACC

Description The Program Counter is restored from the stack and the Accumulator loaded with the specified immediate data. Program execution continues at the restored address.

Operation Program Counter ← Stack
ACC ← x

Affected flag(s) None

RETI Return from interrupt

Description The Program Counter is restored from the stack and the interrupts are re-enabled by setting the EMI bit. EMI is the master interrupt global enable bit. If an interrupt was pending when the RETI instruction is executed, the pending Interrupt routine will be processed before returning to the main program.

Operation Program Counter ← Stack
EMI ← 1

Affected flag(s) None

RL [m] Rotate Data Memory left

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

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

Affected flag(s) None

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

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

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

Affected flag(s) None

RLC [m] Rotate Data Memory left through Carry

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

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

Affected flag(s) C

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

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

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

Affected flag(s) C

RR [m] Rotate Data Memory right

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

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

Affected flag(s) None

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

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

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

Affected flag(s) None

RRC [m] Rotate Data Memory right through Carry

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

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

Affected flag(s) C

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

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

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

Affected flag(s): C

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

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

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

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

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

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

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

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

SDZ [m] Skip if decrement Data Memory is 0

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

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

Affected flag(s): None

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

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

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

Affected flag(s): None

SET [m] Set Data Memory

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

Operation: [m] ← FFH

Affected flag(s): None

SET [m].i Set bit of Data Memory

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

Operation: [m].i ← 1

Affected flag(s): None

SIZ [m] Skip if increment Data Memory is 0

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

Operation [m] ← [m] + 1
Skip if [m]=0

Affected flag(s) None

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

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

Operation ACC ← [m] + 1
Skip if ACC=0

Affected flag(s) None

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

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

Operation Skip if [m].i ≠ 0

Affected flag(s) None

SUB A,[m] Subtract Data Memory from ACC

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

Operation ACC ← ACC − [m]

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

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

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

Operation [m] ← ACC − [m]

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

SUB A,x Subtract immediate data from ACC

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

Operation ACC ← ACC − x

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

SWAP [m] Swap nibbles of Data Memory

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

Operation [m].3~[m].0 ↔ [m].7~[m].4

Affected flag(s) None

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

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

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

Affected flag(s) None

SZ [m] Skip if Data Memory is 0

Description 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

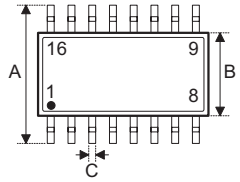
Package Information

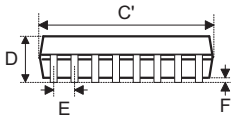
Note that the package information provided here is for consultation purposes only. As this information may be updated at regular intervals users are reminded to consult the Holtek website for the latest version of the Package/Carton Information.

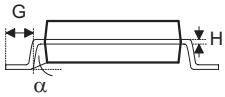
Additional supplementary information with regard to packaging is listed below. Click on the relevant section to be transferred to the relevant website page.

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

16-pin NSOP (150mil) Outline Dimensions







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

Symbol	Dimensions in mm		
	Min.	Nom.	Max.
A	—	6.000 BSC	—
B	—	3.900 BSC	—
C	0.31	—	0.51
C'	—	9.900 BSC	—
D	—	—	1.75
E	—	1.270 BSC	—
F	0.10	—	0.25
G	0.40	—	1.27
H	0.10	—	0.25
α	0°	—	8°