# A/D Flash MCU with Linear Charger

# BP45F1132

Revision: V1.00    Date: November 11, 2020

# 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
- Power down and wake-up functions to reduce power consumption
- Oscillator types
  - Internal High Speed 8MHz RC – HIRC
  - Internal Low Speed 32kHz RC – LIRC
- Multi-mode operation: FAST, SLOW, IDLE and SLEEP
- Fully integrated internal oscillators require no external components
- All instructions executed in one or two instruction cycles
- Table read instructions
- 61 powerful instructions
- 4-level subroutine nesting
- Bit manipulation instruction

### Peripheral Features

- Flash Program Memory: 2K×15
- RAM Data Memory: 128×8
- True EEPROM Memory: 32×8
- Watchdog Timer function
- Up to 18 bidirectional I/O lines
- Programmable I/O source current for LED applications
- Two external interrupt line shared with I/O pins
- One 8-bit programmable Timer/Event Counter with overflow interrupt and prescaler
- Two pairs of 8-bit PWM complementary outputs shared with I/O lines
- Single Time Base function for generation of fixed time interrupt signal
- 4 external channel 12-bit resolution A/D converter with Internal Reference Voltage $V_R$
- One operational amplifier
- Linear charger function
- Over temperature detector function
- Low Voltage Reset function
- Package types: 16-pin NSOP-EP, 24-pin SSOP-EP/QFN

## General Description

The device is a Flash Memory A/D type 8-bit high performance RISC architecture microcontroller with integrated Linear Charger.

For memory features, the Flash Memory offers users the convenience of multi-programming features. Other memory includes an area of RAM Data Memory as well as an area of true EEPROM memory for storage of non-volatile data such as serial numbers, calibration data etc.

Analog features include a multi-channel 12-bit A/D converter, an operational amplifier and an over temperature detector. With regard to internal timer, a single extremely flexible Timer/Event Counter provides timing, event counting and pulse wide measurement functions. Two Pulse Width Modulators provide two pairs of 8-bit PWM complementary outputs. Protective features such as an internal Watchdog Timer and Low Voltage Reset coupled with excellent noise immunity and ESD protection ensure that reliable operation is maintained in hostile electrical environments.

The device also includes fully integrated high and low speed oscillators which require no external components for their implementation. The ability to operate and switch dynamically between a range of operating modes using different clock sources gives users the ability to optimise microcontroller operation and minimise power consumption.

The inclusion of flexible I/O programming features, Time Base function, along with many other features ensure that the device will find excellent use in lithium-ion battery powered handheld products.

## Block Diagram

## Pin Assignment

| | | | |
|---|---|---|---|
| PA4/VREF | 1 | 16 | PA3/INT1/AN1 |
| PA5/AN2 | 2 | 15 | PA1/INT0/AN0 |
| VSS | 3 | 14 | PB4/OPINP/OPINN0 |
| VDD | 4 | 13 | PA2/ICPCK/OCDSCK |
| PB2/PWM1 | 5 | 12 | PA0/ICPDA/OCDSDA |
| PA6/AN3 | 6 | 11 | VBAT |
| PB5/OPINN1 | 7 | 10 | LIN_VCC |
| PB0/PWM0 | 8 | 9 | LIN_VSS |

**BP45F1132/BP45V1132**
**16 NSOP-EP-A**

| | | | |
|---|---|---|---|
| PA1/INT0/AN0 | 1 | 24 | PB0/PWM0 |
| PA3/INT1/AN1 | 2 | 23 | PB1/PWM0B |
| PA4/VREF | 3 | 22 | PB3/PWM1B |
| PA5/AN2 | 4 | 21 | PA7/TC |
| VSS | 5 | 20 | N.C. |
| VDD | 6 | 19 | PA0/ICPDA/OCDSDA |
| PB2/PWM1 | 7 | 18 | PA2/ICPCK/OCDSCK |
| PA6/AN3 | 8 | 17 | PC1 |
| PB7/OPARO | 9 | 16 | PC0 |
| PB6/OPINN0/OPINP | 10 | 15 | VBAT |
| PB5/OPINN1 | 11 | 14 | LIN_VCC |
| PB4/OPINP/IPINN0 | 12 | 13 | LIN_VSS |

**BP45F1132/BP45V1132**
**24 SSOP-EP-A**



**BP45F1132**
**BP45V1132**
**24 QFN-A**

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

2. The OCDSDA and OCDSCK pins are supplied as OCDS dedicated pins and as such only available for the BP45V1132 device which is the OCDS EV chip for the BP45F1132 device.

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

## Pin Description

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

| Pin Name | Function | OPT | I/T | O/T | Description |
|---|---|---|---|---|---|
| PA0/ICPDA/OCDSDA | PA0 | PAPU PAWU | ST | CMOS | General purpose I/O. Register enabled pull-high and wake-up |
| | ICPDA | — | ST | CMOS | ICP data/address |
| | OCDSDA | — | ST | CMOS | OCDS data/address, for EV chip only |
| PA1/INT0/AN0 | PA1 | PAPU PAWU PAS0 | ST | CMOS | General purpose I/O. Register enabled pull-high and wake-up |
| | INT0 | PAS0 INTC0 INTEG | ST | — | External interrupt input 0 |
| | AN0 | PAS0 | AN | — | A/D Converter external input channel 0 |
| PA2/ICPCK/OCDSCK | PA2 | PAPU PAWU | ST | CMOS | General purpose I/O. Register enabled pull-high and wake-up |
| | ICPCK | — | ST | — | ICP clock pin |
| | OCDSCK | — | ST | — | OCDS clock pin, for EV chip only |
| PA3/INT1/AN1 | PA3 | PAPU PAWU PAS0 | ST | CMOS | General purpose I/O. Register enabled pull-high and wake-up |
| | INT1 | PAS0 INTC1 INTEG | ST | — | External interrupt input 1 |
| | AN1 | PAS0 | AN | — | A/D Converter external input channel 1 |
| PA4/VREF | PA4 | PAPU PAWU PAS1 | ST | CMOS | General purpose I/O. Register enabled pull-high and wake-up |
| | VREF | PAS1 | AN | — | A/D Converter external reference input |
| PA5/AN2 | PA5 | PAPU PAWU PAS1 | ST | CMOS | General purpose I/O. Register enabled pull-high and wake-up |
| | AN2 | PAS1 | AN | — | A/D Converter external input channel 2 |
| PA6/AN3 | PA6 | PAPU PAWU PAS1 | ST | CMOS | General purpose I/O. Register enabled pull-high and wake-up |
| | AN3 | PAS1 | AN | — | A/D Converter external input channel 3 |
| PA7/TC | PA7 | PAPU PAWU | ST | CMOS | General purpose I/O. Register enabled pull-high and wake-up |
| | TC | — | ST | — | Timer/Event Counter clock input |
| PB0/PWM0 | PB0 | PBPU PBS0 | ST | CMOS | General purpose I/O. Register enabled pull-high |
| | PWM0 | PBS0 | — | CMOS | PWM0 signal output |
| PB1/PWM0B | PB1 | PBPU PBS0 | ST | CMOS | General purpose I/O. Register enabled pull-high |
| | PWM0B | PBS0 | — | CMOS | PWM0 signal inverting output |
| PB2/PWM1 | PB2 | PBPU PBS0 | ST | CMOS | General purpose I/O. Register enabled pull-high |
| | PWM1 | PBS0 | — | CMOS | PWM1 signal output |

| Pin Name | Function | OPT | I/T | O/T | Description |
|---|---|---|---|---|---|
| PB3/PWM1B | PB3 | PBPU PBS0 | ST | CMOS | General purpose I/O. Register enabled pull-high |
| | PWM1B | PBS0 | — | CMOS | PWM1 signal inverting output |
| PB4/OPINP/OPINN0 | PB4 | PBPU PBS1 | ST | CMOS | General purpose I/O. Register enabled pull-high |
| | OPINP | PBS1 | AN | — | OPAMP postive intput |
| | OPINN0 | PBS1 | AN | — | OPAMP negative input |
| PB5/OPINN1 | PB5 | PBPU PBS1 | ST | CMOS | General purpose I/O. Register enabled pull-high |
| | OPINN1 | PBS1 | AN | — | OPAMP negative input |
| PB6/OPINN0/OPINP | PB6 | PBPU PBS1 | ST | CMOS | General purpose I/O. Register enabled pull-high |
| | OPINN0 | PBS1 | AN | — | OPAMP negative input |
| | OPINP | PBS1 | AN | — | OPAMP postive intput |
| PB7/OPARO | PB7 | PBPU PBS1 | ST | CMOS | General purpose I/O. Register enabled pull-high |
| | OPARO | PBS1 | — | AN | OPAMP output |
| PC0 | PC0 | PCPU | ST | CMOS | General purpose I/O. Register enabled pull-high |
| PC1 | PC1 | PCPU | ST | CMOS | General purpose I/O. Register enabled pull-high |
| LIN_VCC | LIN_VCC | — | PWR | — | Linear charger positive power supply |
| VBAT | VBAT | — | — | PWR | Linear charger output |
| LIN_VSS | LIN_VSS | — | PWR | — | Linear charger negative power supply |
| VDD | VDD | — | PWR | — | Positive power supply |
| VSS | VSS | — | PWR | — | Negative power supply |
| N.C. | N.C. | — | — | — | Not connected |

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

$I_{OL}$ Total ...................................................................................................................... 80mA

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

Total Power Dissipation ............................................................................................. 500mW

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

# D.C. Characteristics

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

## Operating Voltage Characteristics

Ta=-40°C~85°C

| Symbol | Parameter | Test Conditions | Min. | Typ. | Max. | Unit |
|---|---|---|---|---|---|---|
| $V_{DD}$ | Operating Voltage – HIRC | $f_{SYS}=f_{HIRC}$=8MHz | 2.2 | — | 5.5 | V |
| | Operating Voltage – LIRC | $f_{SYS}=f_{LIRC}$=32kHz | 2.2 | — | 5.5 | V |

## Operating Current Characteristics

Ta=-40°C~85°C

| Symbol | Operating Mode | Test Conditions $V_{DD}$ | Conditions | Min. | Typ. | Max. | Unit |
|---|---|---|---|---|---|---|---|
| $I_{DD}$ | SLOW Mode – LIRC | 2.2V | $f_{SYS}$=32kHz | — | 8 | 16 | µA |
| | | 3V | | — | 10 | 20 | |
| | | 5V | | — | 30 | 50 | |
| | FAST Mode – HIRC | 2.2V | $f_{SYS}$=8MHz | — | 0.6 | 1.0 | mA |
| | | 3V | | — | 0.8 | 1.2 | |
| | | 5V | | — | 1.6 | 2.4 | |

Note: When using the characteristic table data, the following notes should be taken into consideration:
1. Any digital inputs are 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.

## Standby Current Characteristics

Ta=25°C, unless otherwise specified

| Symbol | Standby Mode | Test Conditions $V_{DD}$ | Conditions | Min. | Typ. | Max. | Max. @85°C | Unit |
|---|---|---|---|---|---|---|---|---|
| $I_{STB}$ | SLEEP Mode | 2.2V | WDT off | — | 0.08 | 0.12 | 1.40 | µA |
| | | 3V | | — | 0.08 | 0.12 | 1.40 | |
| | | 5V | | — | 0.15 | 0.29 | 2.20 | |
| | IDLE0 Mode – LIRC | 2.2V | $f_{SUB}$ on | — | 2.4 | 4.0 | 4.8 | µA |
| | | 3V | | — | 3 | 5 | 6 | |
| | | 5V | | — | 5 | 10 | 12 | |
| | IDLE1 Mode – HIRC | 2.2V | $f_{SUB}$ on, $f_{SYS}$=8MHz | — | 288 | 400 | 480 | µA |
| | | 3V | | — | 360 | 500 | 600 | |
| | | 5V | | — | 600 | 800 | 960 | |

Note: When using the characteristic table data, the following notes should be taken into consideration:
1. Any digital inputs are 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 Standby Current values are taken after a HALT instruction execution thus stopping all instruction execution.

## A.C. Characteristics

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

### High Speed Internal Oscillator – HIRC – Frequency Accuracy

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

| Symbol | Parameter | Test Conditions | | Min. | Typ. | Max. | Unit |
|--------|-----------|-----|-----|------|------|------|------|
| | | $V_{DD}$ | Temp. | | | | |
| $f_{HIRC}$ | 8MHz Writer Trimmed HIRC Frequency | 3V/5V | 25°C | -1% | 8 | +1% | MHz |
| | | | -40°C~85°C | -3.5% | 8 | +3.5% | |
| | | 2.2V~5.5V | 25°C | -2.5% | 8 | +2.5% | |
| | | | -40°C~85°C | -3.5% | 8 | +3.5% | |

Note: 1. The 3V/5V values for $V_{DD}$ are provided as these are the two selectable fixed voltages at which the HIRC frequency is trimmed by the writer.
2. The row below the 3V/5V trim voltage row is provided to show the values for the full $V_{DD}$ range operating voltage. It is recommended that the trim voltage is fixed at 3V for application voltage ranges from 2.2V to 3.6V and fixed at 5V for application voltage ranges from 3.3V to 5.5V.

### Low Speed Internal Oscillator Characteristics – LIRC

| Symbol | Parameter | Test Conditions | | Min. | Typ. | Max. | Unit |
|--------|-----------|-----|-----|------|------|------|------|
| | | $V_{DD}$ | Temp. | | | | |
| $f_{LIRC}$ | LIRC Frequency | 5V | 25°C | 25.6 | 32 | 38.4 | kHz |
| | | 2.2V~5.5V | 25°C | 12.8 | 32 | 41.6 | |
| | | | -40°C~85°C | 8 | 32 | 60 | |
| $t_{START}$ | LIRC Start-up Time | — | 25°C | — | — | 100 | μs |

### Operating Frequency Characteristic Curves
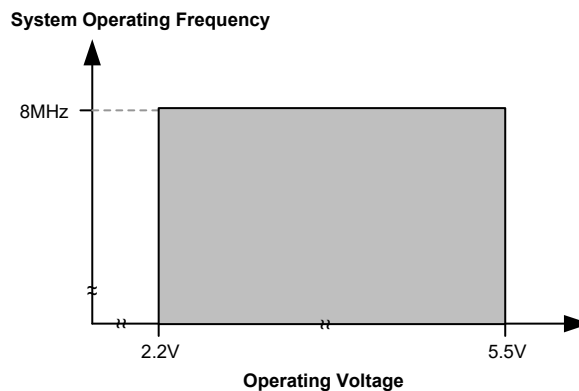
### System Start Up Time Characteristics

Ta=-40°C~85°C

| Symbol | Parameter | Test Conditions | Min. | Typ. | Max. | Unit |
|---|---|---|---|---|---|---|
| $t_{SST}$ | System Start-up Time (Wake-up from Condition where $f_{SYS}$ is off) | $f_{SYS}=f_H\sim f_H/64$, $f_H=f_{HIRC}$ | — | 16 | — | $t_{HIRC}$ |
| | | $f_{SYS}=f_{SUB}=f_{LIRC}$ | — | 2 | — | $t_{LIRC}$ |
| | System Start-up Time (Wake-up from Condition where $f_{SYS}$ is on) | $f_{SYS}=f_H\sim f_H/64$, $f_H=f_{HIRC}$ | — | 2 | 3 | $t_H$ |
| | | $f_{SYS}=f_{SUB}=f_{LIRC}$ | — | 2 | 3 | $t_{SUB}$ |
| | System Speed Switch Time (FAST to SLOW Mode or SLOW to FAST Mode) | $f_{HIRC}$ switches from off → on | 14 | 16 | 18 | $t_{HIRC}$ |
| $t_{RSTD}$ | System Reset Delay Time (Reset source from Power-on reset or LVR Hardware Reset) | $RR_{POR}$=5V/ms | 25 | 50 | 150 | ms |
| | System Reset Delay Time (LVRC/WDTC Register Software Reset) | — | | | | |
| | System Reset Delay Time (WDT Overflow Reset) | — | 8.3 | 16.7 | 50.0 | |
| $t_{SRESET}$ | Minimum Software Reset Width to Reset | — | 45 | 90 | 375 | μs |

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

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

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

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

## Input/Output Characteristics

Ta=-40°C~85°C

| Symbol | Parameter | Test Conditions | | Min. | Typ. | Max. | Unit |
|---|---|---|---|---|---|---|---|
| | | $V_{DD}$ | Conditions | | | | |
| $V_{IL}$ | Input Low Voltage for I/O Ports | 5V | — | 0 | — | 1.5 | V |
| | | — | | 0 | — | $0.2V_{DD}$ | |
| $V_{IH}$ | Input High Voltage for I/O Ports | 5V | — | 3.5 | — | 5.0 | V |
| | | — | | $0.8V_{DD}$ | — | $V_{DD}$ | |
| $I_{OL}$ | Sink Current for I/O Ports | 3V | $V_{OL}$=0.1$V_{DD}$ | 16 | 32 | — | mA |
| | | 5V | | 32 | 65 | — | |
| $I_{OH}$ | Source Current for I/O Ports (except PB2) | 3V | $V_{OH}$=0.9$V_{DD}$, SLEDCn[m+1, m]=00B (n=0 or 1; m=0, 2, 4 or 6) | -0.7 | -1.5 | — | mA |
| | | 5V | | -1.5 | -2.9 | — | |
| | | 3V | $V_{OH}$=0.9$V_{DD}$, SLEDCn[m+1, m]=01B (n=0 or 1; m=0, 2, 4 or 6) | -1.3 | -2.5 | — | |
| | | 5V | | -2.5 | -5.1 | — | |
| | | 3V | $V_{OH}$=0.9$V_{DD}$, SLEDCn[m+1, m]=10B (n=0 or 1; m=0, 2, 4 or 6) | -1.8 | -3.6 | — | |
| | | 5V | | -3.6 | -7.3 | — | |
| | | 3V | $V_{OH}$=0.9$V_{DD}$, SLEDCn[m+1, m]=11B (n=0 or 1; m=0, 2, 4 or 6) | -4 | -8 | — | |
| | | 5V | | -8 | -16 | — | |

| Symbol | Parameter | Test Conditions | | Min. | Typ. | Max. | Unit |
|---|---|---|---|---|---|---|---|
| | | $V_{DD}$ | Conditions | | | | |
| $I_{OH}$ | Source Current for PB2 | 3V | $V_{OH}$=0.9$V_{DD}$, SLEDCn[m+1, m]=00B (n=2; m=0) | -12.5 | -25.0 | — | mA |
| | | 5V | | -25 | -50 | — | |
| | | 3V | $V_{OH}$=0.9$V_{DD}$, SLEDCn[m+1, m]=01B (n=2; m=0) | -18.75 | -37.50 | — | |
| | | 5V | | -37.5 | -75.0 | — | |
| | | 3V | $V_{OH}$=0.9$V_{DD}$, SLEDCn[m+1, m]=10B (n=2; m=0) | -25 | -50 | — | |
| | | 5V | | -50 | -100 | — | |
| | | 3V | $V_{OH}$=0.9$V_{DD}$, SLEDCn[m+1, m]=11B (n=2; m=0) | -31.2 | -62.5 | — | |
| | | 5V | | -62.5 | -125.0 | — | |
| $R_{PH}$ | Pull-high Resistance for I/O Ports [Note] | 3V | — | 20 | 60 | 100 | kΩ |
| | | 5V | | 10 | 30 | 50 | |
| $I_{LEAK}$ | Input Leakage Current | 5V | $V_{IN}$=$V_{DD}$ or $V_{IN}$=$V_{SS}$ | — | — | ±1 | μA |
| $t_{TC}$ | TC Input Pin Minimum Pulse Width | — | — | 25 | — | — | μs |
| $t_{INT}$ | Interrupt Pin 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.

## Memory Characteristics

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

| Symbol | Parameter | Test Conditions | | Min. | Typ. | Max. | Unit |
|---|---|---|---|---|---|---|---|
| | | $V_{DD}$ | Conditions | | | | |
| **Flash Program Memory** | | | | | | | |
| $t_{DEW}$ | Erase/Write Cycle Time | 5.0V | — | — | 2 | 3 | ms |
| $I_{DDPGM}$ | Programming/Erase Current on $V_{DD}$ | 5.0V | — | — | — | 5 | mA |
| $E_P$ | Cell Endurance | — | — | 10K | — | — | E/W |
| $t_{RETD}$ | Data Retention Time | — | Ta=25°C | — | 40 | — | Year |
| **Data EEPROM Memory** | | | | | | | |
| $V_{DD}$ | Operating Voltage for Read | — | — | 2.2 | — | 5.5 | V |
| | Operating Voltage for Write | — | — | 2.2 | — | 5.5 | |
| $t_{EERD}$ | Read Time | — | — | — | — | 4 | $t_{SYS}$ |
| $t_{EEWR}$ | Write Time | 3.0V~5.5V | — | — | 5 | 11 | ms |
| $E_P$ | Cell Endurance | — | — | 100K | — | — | E/W |
| $t_{RETD}$ | Data Retention Time | — | Ta=25°C | — | 40 | — | Year |
| **RAM Data Memory** | | | | | | | |
| $V_{DR}$ | RAM Data Retention Voltage | — | — | 1 | — | — | V |

## LVR 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 |
| $V_{LVR}$ | Low Voltage Reset Voltage | — | LVR enable | -5% | 2.1 | +5% | V |
| $I_{LVR}$ | Operating Current | 3V | LVR enable, $V_{LVR}$=2.1V | — | — | 15 | µA |
| | | 5V | | — | 15 | 25 | |
| $t_{LVR}$ | Minimum Low Voltage Width to Reset | — | — | 140 | 600 | 1000 | µs |

## 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 |
| $V_{ADI}$ | Input Voltage | — | — | 0 | — | $V_{REF}$ | V |
| $V_{REF}$ | Reference Voltage | — | — | 2 | — | $V_{DD}$ | V |
| $N_R$ | Resolution | — | — | — | — | 12 | Bit |
| DNL | Differential Non-linearity | — | $V_{REF}=V_{DD}$, $t_{ADCK}$=0.5µs | -3 | — | 3 | LSB |
| INL | Integral Non-linearity | — | $V_{REF}=V_{DD}$, $t_{ADCK}$=0.5µs | -4 | — | 4 | 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 | |
| $t_{ADCK}$ | Clock Period | — | — | 0.5 | — | 10.0 | µs |
| $t_{ON2ST}$ | A/D Converter On-to-Start Time | — | — | 4 | — | — | µs |
| $t_{ADS}$ | Sampling Time | — | — | — | 4 | — | $t_{ADCK}$ |
| $t_{ADC}$ | Conversion Time (Including A/D Sample and Hold Time) | — | — | — | 16 | — | $t_{ADCK}$ |
| GERR | A/D Conversion Gain Error | — | $V_{REF}=V_{DD}$ | -4 | — | 4 | LSB |
| OSRR | A/D Conversion Offset Error | — | $V_{REF}=V_{DD}$ | -4 | — | 4 | LSB |
| $V_R$ | OPA Output Voltage | — | $V_{DD} \geq 2.4976+0.15$(Min.) and $V_{DD} \leq 5.5V$(Max.) | -1% | 2.4976 | +1% | V |
| R | The Sum of R1 and R2 | 3V | — | 2 | 4 | 6 | kΩ |
| | | 5V | | 2 | 4 | 6 | |
| RR | The ratio of R1/R2 | 3V | — | -1% | 1:1 | +1% | — |
| | | 5V | | -1% | 1:1 | +1% | |

## Linear Charger Electrical Characteristics

$V_{CC}$=5V, Ta=-40°C~85°C, unless otherwise specified

| Symbol | Parameter | Test Conditions | | Min. | Typ. | Max. | Unit |
|---|---|---|---|---|---|---|---|
| | | $V_{CC}$ | Conditions | | | | |
| $V_{IN}$ | Input Voltage | — | — | — | — | 12 | V |
| | | | $V_{BAT}$<4.0V | $V_{UVLO}$ | 5 | $V_{OVP}$ | |
| | | | $V_{BAT}$>4.0V | $V_{BAT}$+0.1 | 5 | $V_{OVP}$ | |
| $V_{CV}$ | $V_{BAT}$ in Constant Voltage Mode | — | — | -1% | 4.2 | 1% | V |
| $V_{UVLO}$ | Under Voltage Lockout | — | — | -3% | 4 | +3% | V |
| $V_{OVP}$ | Over Voltage Protection Voltage | — | — | -3% | 6 | +3% | V |
| $V_{SCC}$ | Trickle Current Mode Threshold Voltage | — | — | -3% | 3 | +3% | V |
| $V_{Recharge}$ | Recharging Threshold Voltage | — | — | -3% | 4.0 | +3% | V |
| $I_{BAT\_SCC}$ | $I_{BAT}$ in Trickle Current Mode | — | $V_{BAT}$<2.2V | — | 20 | — | mA |
| | | | $V_{BAT}$=2.2V~3.0V, $I_{BAT\_CC}$ set by CCSEL[4:0] | — | $I_{BAT\_CC}$/10 | — | |
| $I_{BAT\_CC}$ | $I_{BAT}$ in Constant Current Mode | — | $I_{BAT\_CC}$ set by CCSEL[4:0] | -10% | $I_{BAT\_CC}$ | +10% | mA |
| $I_{BAT\_Cut}$ | Charge Cut-off Current | — | $I_{BAT\_CC}$ set by CCSEL[4:0] | -10% | $I_{BAT\_CC}$/10 | — | mA |
| $I_{VCC\_OP}$ | $I_{VCC}$ Operating Current | — | $V_{CC}$=5V | — | 200 | 250 | µA |
| $I_{BAT\_LK}$ | $I_{BAT}$ Leak Current | — | $V_{BAT}$=4.2V, $V_{CC}$ floating | — | -1 | -2 | µA |
| | | | $V_{BAT}$=4.2V, $V_{CC}$=0V | — | -1 | -2 | |

## Operational Amplifier Electrical Characteristics

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

| Symbol | Parameter | Test Conditions | | Min. | Typ. | Max. | Unit |
|---|---|---|---|---|---|---|---|
| | | $V_{DD}$ | Conditions | | | | |
| $I_{OPA}$ | Additional Current for OPA enable | 3V | No load | — | 150 | 400 | µA |
| | | 5V | | — | 300 | 600 | |
| $V_{OS}$ | Input Offset Voltage | 3V | Without calibration (OPOOF[5:0]=100000B) | -15 | — | +15 | mV |
| | | 5V | | -15 | — | +15 | |
| | | 3V | With calibration | -2 | — | +2 | |
| | | 5V | | -2 | — | +2 | |
| $V_{CM}$ | Common Mode Voltage Range | 5V | Ta=25°C | $V_{SS}$+0.01 | — | 4.12 | V |
| | | | Ta=-40°C~85°C | $V_{SS}$+0.13 | — | 4.02 | |
| $V_{OR}$ | Maximum Output Voltage Range | 5V | — | $V_{SS}$+0.1 | — | $V_{DD}$-0.1 | V |
| $R_{OPAR1}$ | Value of OPAR1 | 5V | — | 7.5 | 10 | 12.5 | kΩ |
| SR | Slew Rate | 5V | No load | 0.6 | 1.8 | — | V/µs |
| GBW | Gain Bandwidth | 5V | $R_{LOAD}$=1MΩ, $C_{LOAD}$=100pF | 600 | 2200 | — | kHz |
| 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 | — | |
| $I_{SOURCE}$ | Output Source Current | 3V | $V_{OH}$=2.7V | 0.6 | 1.0 | — | mA |
| | | 5V | $V_{OH}$=4.5V | 1.4 | 2.4 | — | |
| Ga | PGA Gain Accuracy [Note] | 3V | Relative gain | -5 | — | 5 | % |
| | | 5V | Relative gain | -5 | — | 5 | |

Note: The PGA gain accuracy is guaranted only when the PGA output voltage meets the $V_{OR}$ specification.

## Over Temperature Detector Electrical Characteristics

Ta=-40°C~85°C

| Symbol | Parameter | Test Conditions | | Min. | Typ. | Max. | Unit |
|---|---|---|---|---|---|---|---|
| | | $V_{DD}$ | Conditions | | | | |
| $V_{DD}$ | Operating Voltage | — | — | 2.7 | — | 5.5 | V |
| $T_{OTT}$ | Over Temperature Threshold | — | — | 95 | 110 | 125 | °C |
| $T_{HR}$ | Thermal Release Temperature | — | — | 70 | 85 | 100 | °C |
| $T_{OTW}$ | Over Temperature Window | — | — | 20 | 25 | 30 | °C |
| $I_{OP}$ | Operating Current | 5V | OTDEN=1 | — | 150 | 380 | µA |
| $I_{STB}$ | Standby Current | 5V | OTDEN=0 | — | 0.1 | 1.0 | µA |
| $t_{OTD}$ | Minimum Over Temperature Width to Interrupt | — | — | 120 | 240 | 480 | µs |

## Power-on Reset Characteristics

Ta=-40°C~85°C

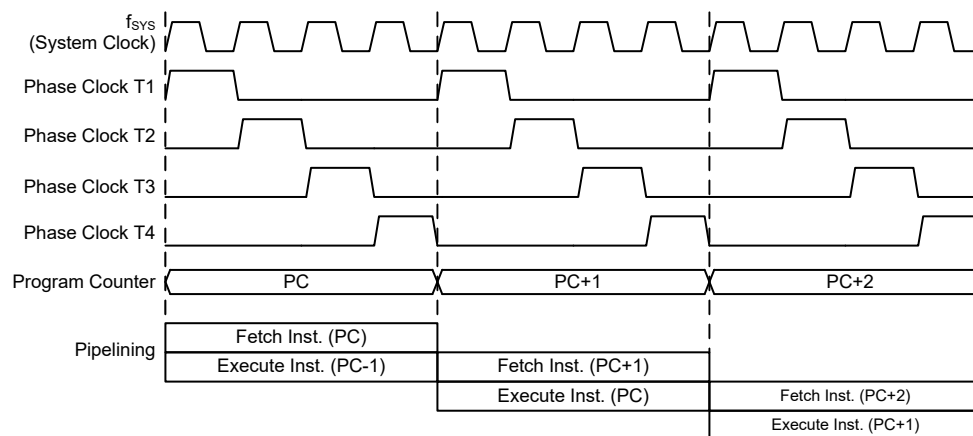| Symbol | Parameter | Test Conditions | | Min. | Typ. | Max. | Unit |
|---|---|---|---|---|---|---|---|
| | | $V_{DD}$ | Conditions | | | | |
| $V_{POR}$ | $V_{DD}$ Start Voltage to Ensure Power-on Reset | — | — | — | — | 100 | mV |
| $RR_{POR}$ | $V_{DD}$ Rising Rate to Ensure Power-on Reset | — |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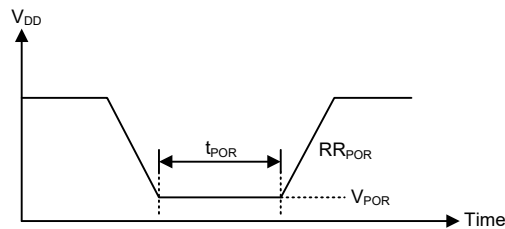
## System Architecture

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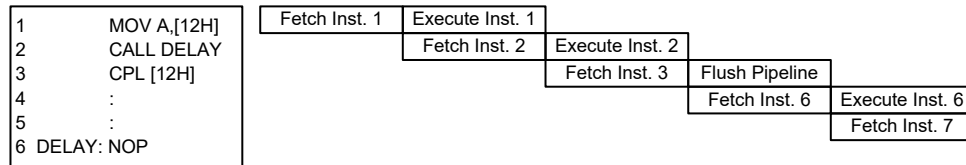
### Clocking and Pipelining

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

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



**System Clocking and Pipelining**

```
1         MOV A,[12H]
2         CALL DELAY
3         CPL [12H]
4         :
5         :
6 DELAY: NOP
```

| Fetch Inst. 1 | Execute Inst. 1 |
| Fetch Inst. 2 | Execute Inst. 2 |
| Fetch Inst. 3 | Flush Pipeline |
| Fetch Inst. 6 | Execute Inst. 6 |
| Fetch Inst. 7 |

**Instruction Fetching**

### Program Counter

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

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

| Program Counter | |
|---|---|
| **High Byte** | **Low Byte (PCL)** |
| PC10~PC8 | PCL7~PCL0 |

**Program Counter**

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

### Stack

This is a special part of the memory which is used to save the contents of the Program Counter only. The stack is organized into multiple 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.
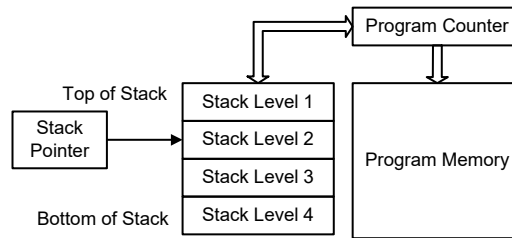
### Arithmetic and Logic Unit – ALU

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 2K×15 bits. The Program Memory is addressed by the Program Counter and also contains data, table information and interrupt entries. Table data, which can be setup in any location within the Program Memory, is addressed by a separate table pointer register.
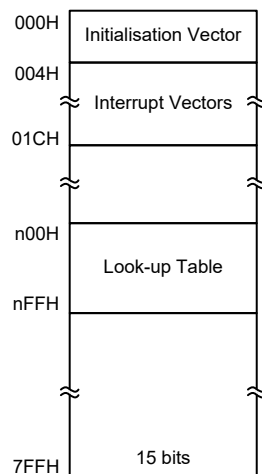
**Program Memory Structure**

## 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 setup 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]" instruction. When the instruction is executed, the lower order table byte from the Program Memory will be transferred to the user defined Data Memory register [m] as specified in the instruction. The higher order table data byte from the Program Memory will be transferred to the TBLH special register.

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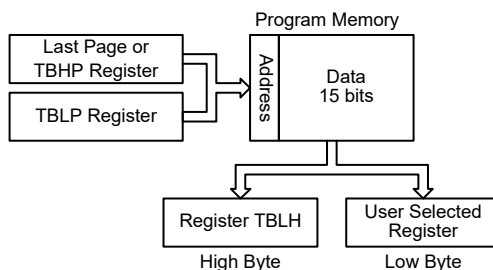


## Table Program Example

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 "700H" which refers to the start address of the last page within the 2K Program Memory of the device. The table pointer low byte register is setup here to have an initial value of "06H". This will ensure that the first data read from

the data table will be at the Program Memory address "706H" or 6 locations after the start of the last page. Note that the value for the table pointer is referenced to the specific address specified by TBLP and TBHP 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 can be restored, care should be taken to ensure its protection if both the main routine and Interrupt Service Routine use table read instructions. If using the table read instructions, the Interrupt Service Routines may change the value of the TBLH and subsequently cause errors if used again by the main routine. As a rule, it is recommended that simultaneous use of the table read instructions should be avoided. However, in situations where simultaneous use cannot be avoided, the interrupts should be disabled prior to the execution of any main routine table-read instructions. Note that all table related instructions require two instruction cycles to complete their operation.

**Table Read Program Example**

```
tempreg1 db ?   ; temporary register #1
tempreg2 db ?   ; temporary register #2
:
:
mov a,06h       ; initialise low table pointer - note that this address is referenced
mov tblp,a      ; to the last page or the page that tbhp pointed
mov a,07h       ; initialise high table pointer
mov tbhp,a
:
:
tabrd tempreg1  ; transfers value in table referenced by table pointer data at program
                ; memory address "706H" transferred to tempreg1 and TBLH
dec tblp        ; reduce value of table pointer by one
tabrd tempreg2  ; transfers value in table referenced by table pointer
                ; data at program memory address "705H" transferred to tempreg2 and TBLH
                ; in this example the data "1AH" is transferred to tempreg1 and data "0FH"
                ; to register tempreg2 and the data "00H" is transferred to TBLH
:
:
org 700h        ; sets initial address of program memory
dc 00Ah, 00Bh, 00Ch, 00Dh, 00Eh, 00Fh, 01Ah, 01Bh
:
:
```

## In Circuit Programming – ICP

The provision of Flash Type Program Memory provides the user with a means of convenient and easy upgrades and modifications to their programs on the same device. As an additional convenience, a means of programming the microcontroller in-circuit has provided 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.

The Holtek Flash MCU to Writer Programming Pin correspondence table is as follows:

| 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, taking control of the ICPDA and ICPCK pins for data and clock programming purposes. The user must there take care to ensure that no other outputs are connected to these two pins.
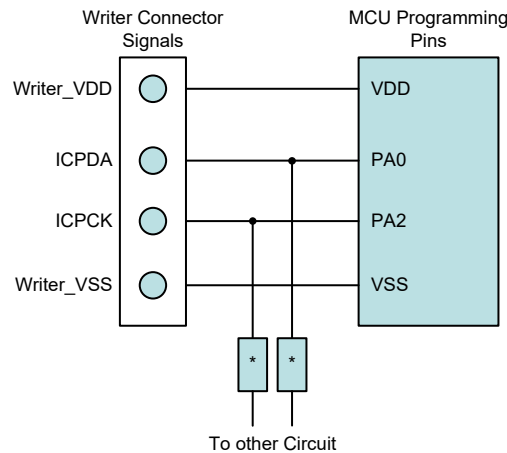


Note: * may be resistor or capacitor. The resistance of * must be greater than 1kΩ or the capacitance of * must be less than 1nF.

## On Chip Debug Support – OCDS

An EV chip exists for the purposes of device emulation. This EV chip device also provides an "On-Chip Debug" function to debug the real MCU device during the development process. The EV chip and the real MCU device are almost functionally compatible except for "On-Chip Debug" function. Users can use the EV chip device to emulate the real chip device behavior by connecting the OCDSDA and OCDSCK pins to the Holtek HT-IDE development tools. The OCDSDA pin is the OCDS Data/Address input/output pin while the OCDSCK pin is the OCDS clock input pin. When users use the EV chip for debugging, other functions which are shared with the OCDSDA and OCDSCK pins in the device will have no effect in the EV chip. 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 Pins | Pin Description |
|---|---|---|
| OCDSDA | OCDSDA | On-Chip Debug Support Data/Address input/output |
| OCDSCK | OCDSCK | On-Chip Debug Support Clock input |
| VDD | VDD | Power Supply |
| VSS | VSS | Ground |

# Data Memory

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

## Structure

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

The Data Memory is subdivided into two banks, all of which are implemented in 8-bit wide RAM. Switching between the different Data Memory banks is achieved by properly setting the Bank Pointer to the correct value. The address range of the Special Purpose Data Memory for the device is from 00H to 3FH while the General Purpose Data Memory address range is from 40H to BFH.

| Special Purpose Data Memory | General Purpose Data Memory | |
|---|---|---|
| **Located Banks** | **Capacity** | **Bank: Address** |
| 0: 00H~3FH<br>1: 40H (EEC only) | 128×8 | 0: 40H~BFH |
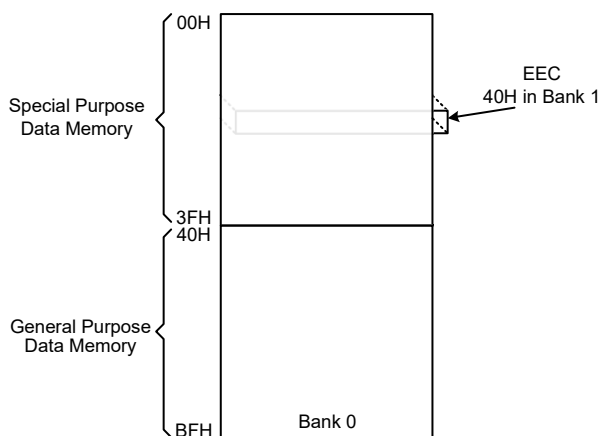


**Data Memory Structure**

## 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".

|  | Bank 0 | Bank 1 |
|---|---|---|
| 00H | IAR0 | |
| 01H | MP0 | |
| 02H | IAR1 | |
| 03H | MP1 | |
| 04H | BP | |
| 05H | ACC | |
| 06H | PCL | |
| 07H | TBLP | |
| 08H | TBLH | |
| 09H | TBHP | |
| 0AH | STATUS | |
| 0BH | | |
| 0CH | | |
| 0DH | | |
| 0EH | | |
| 0FH | RSTFC | |
| 10H | SCC | |
| 11H | HIRCC | |
| 12H | WDTC | |
| 13H | PSCR | |
| 14H | PA | |
| 15H | PAC | |
| 16H | PAPU | |
| 17H | PAWU | |
| 18H | PB | |
| 19H | PBC | |
| 1AH | PBPU | |
| 1BH | PC | |
| 1CH | PCC | |
| 1DH | PCPU | |
| 1EH | EEA | |
| 1FH | EED | |
| 20H | (Reserved) | |
| 21H | TMR | |
| 22H | TMRC | |
| 23H | TBC | |
| 24H | INTC0 | |
| 25H | INTC1 | |
| 26H | INTEG | |
| 27H | PAS0 | |
| 28H | PAS1 | |
| 29H | PBS0 | |
| 2AH | PBS1 | |
| 2BH | SADOL | |
| 2CH | SADOH | |
| 2DH | SADC0 | |
| 2EH | SADC1 | |
| 2FH | SWS0 | |
| 30H | LCS0 | |
| 31H | LCS1 | |
| 32H | OPC0 | |
| 33H | OPC1 | |
| 34H | OPOCAL | |
| 35H | SLEDC0 | |
| 36H | SLEDC1 | |
| 37H | SLEDC2 | |
| 38H | PWM0P | |
| 39H | PWM0D | |
| 3AH | PWM0C | |
| 3BH | PWM1P | |
| 3CH | PWM1D | |
| 3DH | PWM1C | |
| 3EH | LVRC | |
| 3FH | OTDC | |
| 40H | General Purpose Data Memory | EEC |

☐ : Unused, read as 00H

▨ : Reserved, cannot be changed

**Special Purpose Data Memory Structure**

## Special Function Register Description

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

### Indirect Addressing Registers – IAR0, IAR1

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

### Memory Pointers – MP0, MP1

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

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

**Indirect Addressing Program**

```
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 Byte Register – PCL

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

### Look-up Table Registers – TBLP, 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" instructions, allowing for easy table data pointing and reading. TBLH is the location where the high order byte of the table data is stored after a table read data instruction has been executed. Note that the lower order table data byte is transferred to a user defined location.

### Status Register – STATUS

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

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

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

- C is set if an operation results in a carry during an addition operation or if a borrow does not take place during a subtraction operation; otherwise C is cleared. C is also affected by a rotate through carry instruction.

- AC is set if an operation results in a carry out of the low nibbles in addition, or no borrow from the high nibble into the low nibble in subtraction; otherwise AC is cleared.

- Z is set if the result of an arithmetic or logical operation is zero; otherwise Z is cleared.

- OV is set if an operation results in a carry into the highest-order bit but not a carry out of the highest-order bit, or vice versa; otherwise OV is cleared.

- PDF is cleared by a system power-up or executing the "CLR WDT" instruction. PDF is set by executing the "HALT" instruction.

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

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

• **STATUS Register**

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

"x": unknown

Bit 7~6     Unimplemented, read as "0"

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

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

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

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

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

Bit 0     **C**: Carry flag
          0: No carry-out
          1: An operation results in a carry during an addition operation or if a borrow does not take place during a subtraction operation
          The "C" flag is also affected by a rotate through carry instruction.

## Bank Pointer – BP

For this device, the Data Memory is divided into two banks, Bank 0 and Bank 1. Selecting the required Data Memory area is achieved using the Bank Pointer. Bit 0 of the Bank Pointer is used to select Data Memory Banks 0~1.

The Data Memory is initialised to Bank 0 after a reset, except for a WDT time-out reset in the IDLE or SLEEP Mode, in which case, the Data Memory bank remains unaffected. Directly addressing the Data Memory will always result in Bank 0 being accessed irrespective of the value of the Bank Pointer. Accessing data from Bank1 must be implemented using Indirect Addressing.

• **BP Register**

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

Bit 7~1      Unimplemented, read as "0"

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

# EEPROM Data Memory

The device contains an area of internal EEPROM Data Memory. EEPROM 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 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. The process of reading and writing data to the EEPROM memory has been reduced to a very trivial affair.

## EEPROM Data Memory Structure

The EEPROM Data Memory capacity is 32×8 bits for the device. Unlike the Program Memory and RAM Data Memory, the EEPROM Data Memory is not directly mapped into memory space and is therefore not directly addressable in the same way as the other types of memory. Read and Write operations to the EEPROM are carried out in single byte operations using an address register and a data register in Bank 0 and a single control register in Bank 1.

## EEPROM Registers

Three registers control the overall operation of the internal EEPROM Data Memory. These are the address register, EEA, the data register, EED and a single control register, EEC. As both the EEA and EED registers are located in Bank 0, they can be directly accessed in the same way as any other Special Function Register. The EEC register however, being located in Bank 1, can only be read from or written to indirectly using the MP1 Memory Pointer and Indirect Addressing Register, IAR1. Because the EEC control register is located at address 40H in Bank 1, the MP1 Memory Pointer must first be set to the value 40H and the Bank Pointer register, BP, set to the value, 01H, before any operations on the EEC register are executed.

| Register Name | Bit | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| EEA | — | — | — | EEA4 | EEA3 | EEA2 | EEA1 | EEA0 |
| EED | EED7 | EED6 | EED5 | EED4 | EED3 | EED2 | EED1 | EED0 |
| EEC | D7 | — | — | — | WREN | WR | RDEN | RD |

**EEPROM Register List**

HOLTEK

• **EEA Register**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|-----|---|---|---|---|---|---|---|---|
| Name | — | — | — | EEA4 | EEA3 | EEA2 | EEA1 | EEA0 |
| 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        **EEA4~EEA0**: Data EEPROM address bit 4 ~ bit 0

• **EED Register**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|-----|---|---|---|---|---|---|---|---|
| Name | EED7 | EED6 | EED5 | EED4 | EED3 | EED2 | EED1 | EED0 |
| 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        **EED7~EED0**: Data EEPROM data bit 7 ~ bit 0

• **EEC Register**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|-----|---|---|---|---|---|---|---|---|
| Name | D7 | — | — | — | WREN | WR | RDEN | RD |
| R/W | R/W | — | — | — | R/W | R/W | R/W | R/W |
| POR | 0 | — | — | — | 0 | 0 | 0 | 0 |

Bit 7          **D7**: Reserved bit, must be fixed at "0"

Bit 6~4        Unimplemented, read as "0"

Bit 3          **WREN**: Data EEPROM Write Enable
               0: Disable
               1: Enable
               This is the Data EEPROM Write Enable Bit which must be set high before Data
               EEPROM write operations are carried out. Clearing this bit to zero will inhibit Data
               EEPROM write operations.

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

Bit 1          **RDEN**: Data EEPROM Read Enable
               0: Disable
               1: Enable
               This is the Data EEPROM Read Enable Bit which must be set high before Data
               EEPROM read operations are carried out. Clearing this bit to zero will inhibit Data
               EEPROM read operations.

Bit 0          **RD**: EEPROM Read Control
               0: Read cycle has finished
               1: Activate a read cycle
               This is the Data EEPROM Read Control Bit and when set high by the application
               program will activate a read cycle. 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 RDEN has not first been set high.

Note: 1. The WREN, WR, RDEN and RD cannot be set high at the same time in one instruction.
         The WR and RD cannot be set high at the same time.

2. Ensure that the f$_{SUB}$ clock is stable before executing the write operation.

3. Ensure that the write operation is totally complete before changing the contents of the EEPROM related registers.

### Reading Data from the EEPROM

To read data from the EEPROM, the EEPROM address of the data to be read must first be placed in the EEA register. Then the read enable bit, RDEN, in the EEC register must be set high to enable the read function. If the RD bit in the EEC register is now set high, a read cycle will be initiated. Setting the RD bit high will not initiate a read operation if the RDEN bit has not been set. When the read cycle terminates, the RD bit will be automatically cleared to zero, after which the data can be read from the EED register. The data will remain in the EED register until another read or write operation is executed. The application program can poll the RD bit to determine when the data is valid for reading.

### Writing Data to the EEPROM

To write data to the EEPROM, the EEPROM address of the data to be written must first be placed in the EEA register and the data placed in the EED register. To initiate a write cycle, the write enable bit, WREN, in the EEC register must first be set high to enable the write function. After this, the WR bit in the EEC register must be immediately set high to initiate a write cycle. These two instructions must be executed in two consecutive instruction cycles. The global interrupt bit EMI should also first be cleared before implementing any write operations, and then set again after the write cycle has started. Note that setting the WR bit high will not initiate a write cycle if the WREN bit has not been set. As the EEPROM write cycle is controlled using an internal timer whose operation is asynchronous to microcontroller system clock, a certain time will elapse before the data will have been written into the EEPROM. Detecting when the write cycle has finished can be implemented either by polling the WR bit in the EEC register or by using the EEPROM interrupt. When the write cycle terminates, the WR bit will be automatically cleared to zero by the microcontroller, informing the user that the data has been written to the EEPROM. The application program can therefore poll the WR bit to determine when the write cycle has ended.

### Write Protection

Protection against inadvertent write operation is provided in several ways. After the device is powered-on the Write Enable bit in the control register will be cleared preventing any write operations. Also at power-on the Bank Pointer register, BP, will be reset to zero, which means that Data Memory Bank 0 will be selected. As the EEPROM control register is located in Bank 1, this adds a further measure of protection against spurious write operations. During normal program operation, ensuring that the Write Enable bit in the control register is cleared will safeguard against incorrect write operations.

### EEPROM Interrupt

The EEPROM write interrupt is generated when an EEPROM write cycle has ended. The EEPROM interrupt must first be enabled by setting the DEE bit in the relevant interrupt register. When an EEPROM write cycle ends, the DEF request flag will be set. If the global and the EEPROM interrupts are enabled and the stack is not full, a jump to the associated EEPROM Interrupt vector will take place. When the interrupt is serviced the EEPROM interrupt flag, DEF, will be automatically reset and the EMI bit will be automatically cleared to disable other interrupts. More details can be obtained in the Interrupt section.

### Programming Considerations

Care must be taken that data is not inadvertently written to the EEPROM. Protection can be enhanced by ensuring that the Write Enable bit is normally cleared to zero when not writing. Also the Bank Pointer register, BP, could be normally cleared to zero as this would inhibit access to Bank 1 where the EEPROM control register exists. Although certainly not necessary, consideration might be given in the application program to the checking of the validity of new write data by a simple read back process.

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

### Programming Examples

#### Reading data from the EEPROM – polling method

```
MOV  A, EEPROM_ADRES        ; user defined address
MOV  EEA, A
MOV  A, 40H                 ; setup memory pointer MP1
MOV  MP1, A                 ; MP1 points to EEC register
MOV  A, 01H                 ; setup Bank Pointer
MOV  BP, A
SET  IAR1.1                 ; set RDEN bit, enable read operations
SET  IAR1.0                 ; start Read Cycle - set RD bit
BACK:
SZ   IAR1.0                 ; check for read cycle end
JMP  BACK
CLR  IAR1                   ; disable EEPROM read if no more read operations are required
CLR  BP
MOV  A, EED                 ; move read data to register
MOV  READ_DATA, A
```

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

#### Writing Data to the EEPROM – polling method

```
MOV  A, EEPROM_ADRES        ; user defined address
MOV  EEA, A
MOV  A, EEPROM_DATA         ; user defined data
MOV  EED, A
MOV  A, 40H                 ; setup memory pointer MP1
MOV  MP1, A                 ; MP1 points to EEC register
MOV  A, 01H                 ; setup Bank Pointer
MOV  BP, A
CLR  EMI
SET  IAR1.3                 ; set WREN bit, enable write operations
SET  IAR1.2                 ; start Write Cycle - set WR bit - executed immediately
                            ; after setting WREN bit
SET  EMI
BACK:
SZ   IAR1.2                 ; check for write cycle end
JMP  BACK
CLR  BP
```

# Oscillators

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

## Oscillator Overview

In addition to being the source of the main system clock the oscillators also provide clock sources for the Watchdog Timer and Time Base Interrupt. Two fully integrated internal oscillators, requiring no external components, are provided to form a wide range of both fast and slow system oscillators. The higher frequency oscillator provides higher performance but carry with it the disadvantage of higher power requirements, while the opposite is of course true for the lower frequency oscillator. With the capability of dynamically switching between fast and slow system clock, the device has the flexibility to optimize the performance/power ratio, a feature especially important in power sensitive portable applications.

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

**Oscillator Types**

## System Clock Configurations

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



**System Clock Configurations**
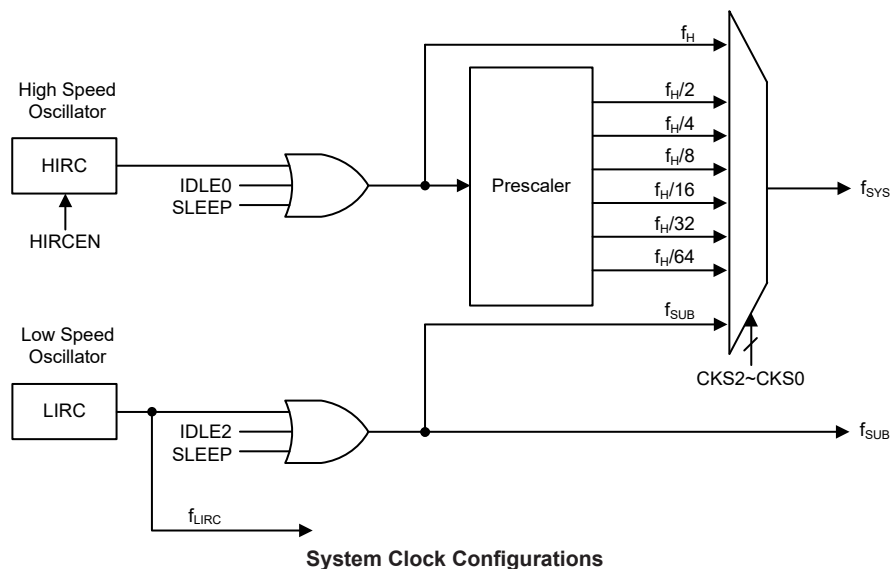
### Internal High Speed RC Oscillator – HIRC

The high speed internal RC oscillator is a fully integrated system oscillator requiring no external components. The internal RC oscillator has a fixed frequency of 8MHz. Device trimming during the manufacturing process and the inclusion of internal frequency compensation circuits are used to ensure that the influence of the power supply voltage, temperature and process variations on the oscillation frequency are minimised.

### Internal 32kHz Oscillator – LIRC

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

## Operating Modes and System Clocks

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

### System Clocks

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

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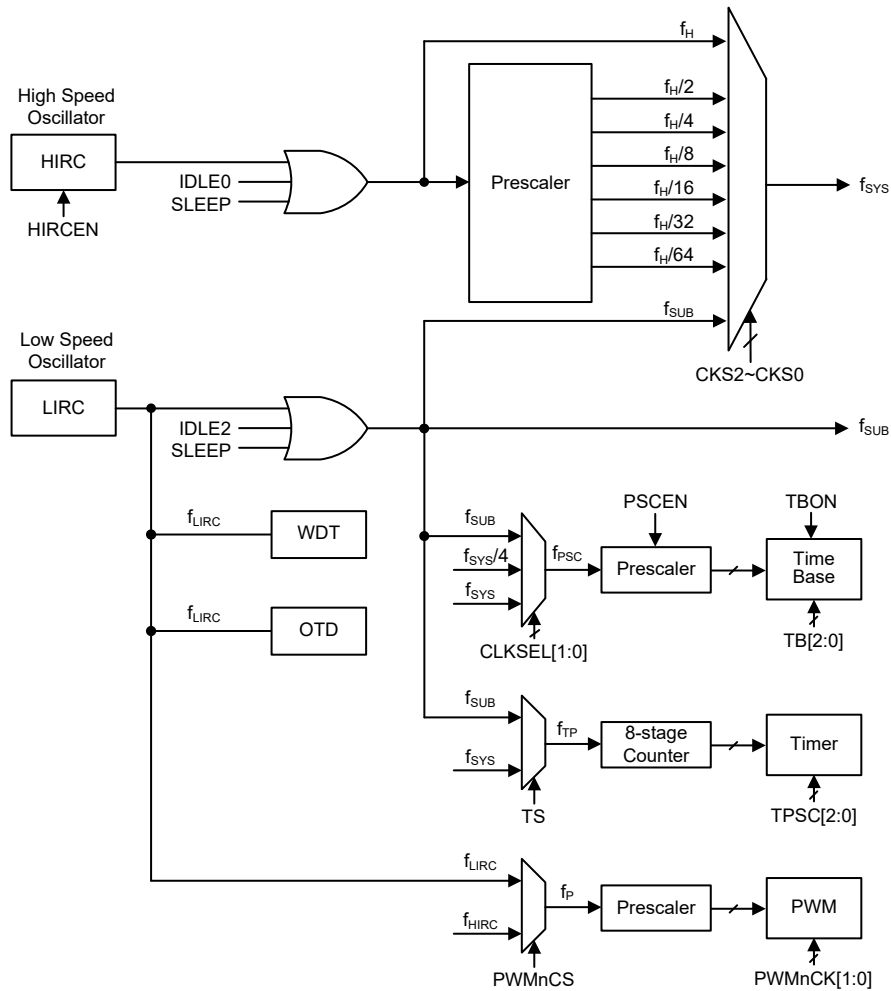
**Device Clock Configurations**

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

## System Operation Modes

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

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

"x": Don't care

Note: 1. The $f_H$ clock will be switched on or off by configuring the corresponding oscillator enable bit in the SLOW mode.
2. The $f_{LIRC}$ clock will be switched on or off which is controlled by the WDT function being enabled or disabled in the SLEEP mode.

### FAST Mode

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

### SLOW Mode

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

### SLEEP Mode

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

### IDLE0 Mode

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

### IDLE1 Mode

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

**IDLE2 Mode**

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

## Control Registers

The SCC and HIRCC registers are used to control the system clock and the HIRC oscillator configurations.

| Register Name | Bit | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| SCC | CKS2 | CKS1 | CKS0 | — | — | — | FHIDEN | FSIDEN |
| HIRCC | — | — | — | — | — | — | HIRCF | HIRCEN |

**System Operating Mode Control Register List**

### • SCC Register

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

Bit 7~5    **CKS2~CKS0**: System clock selection
  000: $f_H$
  001: $f_H/2$
  010: $f_H/4$
  011: $f_H/8$
  100: $f_H/16$
  101: $f_H/32$
  110: $f_H/64$
  111: $f_{SUB}$

  These three bits are used to select which clock is used as the system clock source. In addition to the system clock source directly derived from $f_H$ or $f_{SUB}$, a divided version of the high speed system oscillator can also be chosen as the system clock source.

Bit 4~2    Unimplemented, read as "0"

Bit 1    **FHIDEN**: High Frequency oscillator control when CPU is switched off
  0: Disable
  1: Enable

  This bit is used to control whether the high speed oscillator is activated or stopped when the CPU is switched off by executing a "HALT" instruction.

Bit 0    **FSIDEN**: Low Frequency oscillator control when CPU is switched off
  0: Disable
  1: Enable

  This bit is used to control whether the low speed oscillator is activated or stopped when the CPU is switched off by executing a "HALT" instruction.

• **HIRCC Register**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|-----|---|---|---|---|---|---|---|---|
| Name | — | — | — | — | — | — | HIRCF | HIRCEN |
| R/W | — | — | — | — | — | — | R | R/W |
| POR | — | — | — | — | — | — | 0 | 1 |

Bit 7~2  Unimplemented, read as "0"

Bit 1   **HIRCF**: HIRC oscillator stable flag
     0: HIRC unstable
     1: HIRC stable

     This bit is used to indicate whether the HIRC oscillator is stable or not. When the HIRCEN bit is set to 1 to enable the HIRC oscillator, the HIRCF bit will first be cleared to 0 and then set to 1 after the HIRC oscillator is stable.

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

## Operating Mode Switching

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

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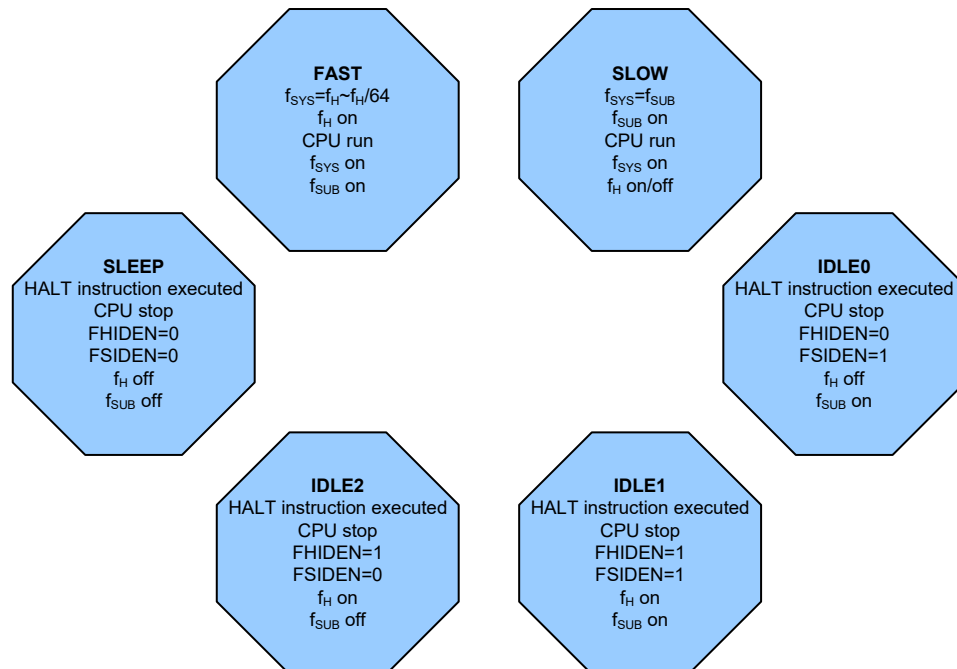
**FAST Mode to SLOW Mode Switching**

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

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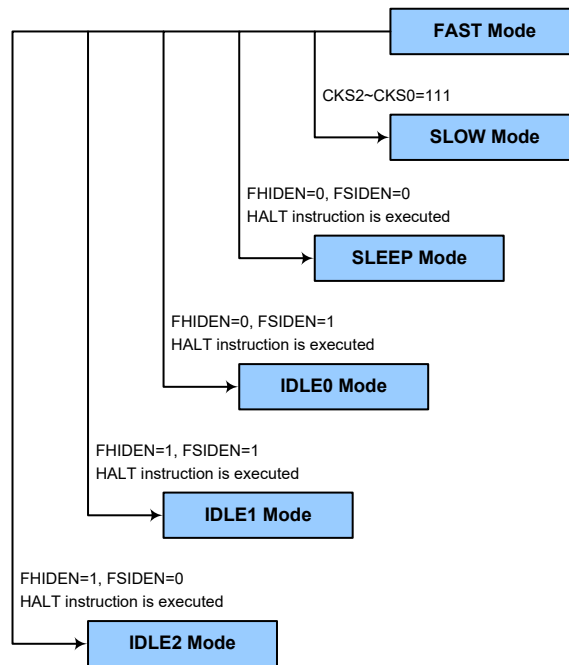


**SLOW Mode to FAST Mode Switching**

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

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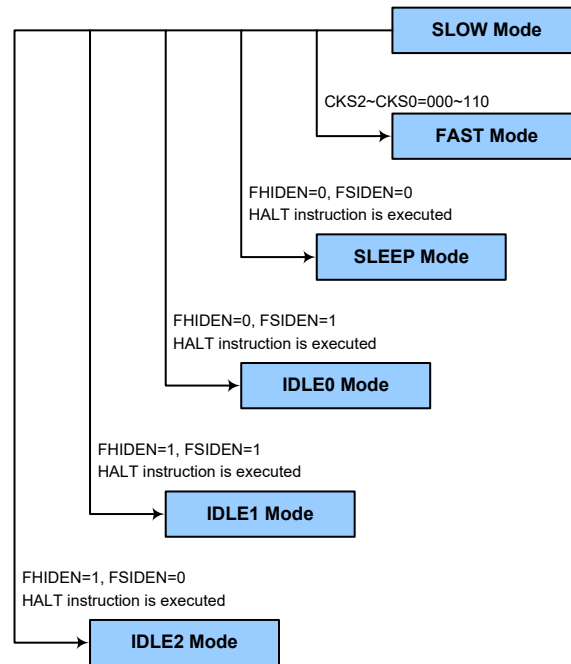
### Entering the SLEEP Mode

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

• The system clock will be stopped and the application program will stop at the "HALT" instruction.

• The Data Memory contents and registers will maintain their present condition.

• The I/O ports will maintain their present conditions.

• In the status register, the Power Down flag, PDF, will be set and the Watchdog time-out flag, TO, will be cleared.

• The WDT will be cleared and resume counting if the WDT function is enabled. If the WDT function is disabled, the WDT will be cleared and then stopped.

### Entering the IDLE0 Mode

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

• The $f_H$ clock will be stopped and the application program will stop at the "HALT" instruction, but the $f_{SUB}$ clock will be on.

• The Data Memory contents and registers will maintain their present condition.

• The I/O ports will maintain their present conditions.

• In the status register, the Power Down flag, PDF, will be set and the Watchdog time-out flag, TO, will be cleared.

• The WDT will be cleared and resume counting if the WDT function is enabled. If the WDT function is disabled, the WDT will be cleared and then stopped.

### Entering the IDLE1 Mode

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

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

- The Data Memory contents and registers will maintain their present condition.

- The I/O ports will maintain their present conditions.

- In the status register, the Power Down flag, PDF, will be set and the Watchdog time-out flag, TO, will be cleared.

- The WDT will be cleared and resume counting if the WDT function is enabled. If the WDT function is disabled, the WDT will be cleared and then stopped.

### Entering the IDLE2 Mode

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

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

- The Data Memory contents and registers will maintain their present condition.

- The I/O ports will maintain their present conditions.

- In the status register, the Power Down flag, PDF, will be set and the Watchdog time-out flag, TO, will be cleared.

- The WDT will be cleared and resume counting if the WDT function is enabled. If the WDT function is disabled, the WDT will be cleared and then stopped.

## Standby Current Considerations

As the main reason for entering the SLEEP or IDLE Mode is to keep the current consumption of the device to as low a value as possible, perhaps only in the order of several micro-amps except in the IDLE1 and IDLE2 Modes, there are other considerations which must also be taken into account by the circuit designer if the power consumption is to be minimised. Special attention must be made to the I/O pins on the device. All high-impedance input pins must be connected to either a fixed high or low level as any floating input pins could create internal oscillations and result in increased current consumption. This also applies to the device which has different package types, as there may be unbonded pins. These pins must either be setup as outputs or if setup as inputs must have pull-high resistors connected.

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

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

#### Wake-up

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

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

- An external falling edge on Port A
- A system interrupt
- A WDT overflow

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

If the system is woken up by a WDT overflow, a Watchdog Timer Time-out reset will be initiated and the TO flag will be set to 1. The TO flag is set high if a WDT time-out occurs, and causes a wake-up that only resets the Program Counter and Stack Pointer, the other flags remain in their original status.

Each pin on Port A can be setup using the PAWU register to permit a negative transition on the pin to wake-up the system. When a pin wake-up occurs, the program will resume execution at the instruction following the "HALT" instruction.

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

## Watchdog Timer

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

### Watchdog Timer Clock Source

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

### Watchdog Timer Control Register

A single register, WDTC, controls the required timeout period as well as the Watchdog Timer enable/disable and MCU reset operation.

• **WDTC Register**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|-----|---|---|---|---|---|---|---|---|
| Name | WE4 | WE3 | WE2 | WE1 | WE0 | WS2 | WS1 | WS0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| POR | 0 | 1 | 0 | 1 | 0 | 0 | 1 | 1 |

Bit 7~3   **WE4~WE0**: WDT function software control
            01010: Enable
            10101: Disable
            Other values: Reset MCU

            When these bits are changed by the environmental noise or software setting to reset
            the microcontroller, the 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: $2^9/f_{LIRC}$
            010: $2^{10}/f_{LIRC}$
            011: $2^{11}/f_{LIRC}$
            100: $2^{12}/f_{LIRC}$
            101: $2^{13}/f_{LIRC}$
            110: $2^{14}/f_{LIRC}$
            111: $2^{15}/f_{LIRC}$

            These three bits determine the division ratio of the watchdog timer source clock, which
            in turn determines the time-out period.

• **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**: WDTC register software reset flag
          0: Not occurred
          1: Occurred
        This bit is set high by the WDTC register software reset and cleared to zero by the
        application program. Note that this bit can be cleared to zero only by the application
        program.

## Watchdog Timer Operation

The Watchdog Timer operates by providing a device reset when its timer overflows. This means
that in the application program and during normal operation the user has to strategically clear the
Watchdog Timer before it overflows to prevent the Watchdog Timer from executing a reset. This is
done using the clear watchdog instruction. If the program malfunctions for whatever reason, jumps
to an unknown location, or enters an endless loop, the clear instruction will not be executed in the
correct manner, in which case the Watchdog Timer will overflow and reset the device. There are
five bits, WE4~WE0, in the WDTC register to offer the Watchdog Timer enable/disable control and
the MCU reset. The WDT function will be disabled when the WE4~WE0 bits are set to a value of

10101B while the WDT function will be enabled if the WE4~WE0 bits are equal to 01010B. If the WE4~WE0 bits are set to any other values, other than 01010B and 10101B, it will reset the device after a delay time, $t_{SRESET}$. After power on these bits will have a value of 01010B.

| WE4~WE0 Bits | WDT Function |
|---|---|
| 10101B | Disable |
| 01010B | Enable |
| Any other values | Reset MCU |

**Watchdog Timer Enable/Disable Control**

Under normal program operation, a Watchdog Timer time-out will initialise a device reset and set the status bit TO high. However, if the system is in the SLEEP or IDLE Mode, when a Watchdog Timer time-out occurs, the TO bit in the status register will be set high and only the Program Counter and Stack Pointer will be reset. Three methods can be adopted to clear the contents of the Watchdog Timer. The first is a WDTC software reset, which means a certain value except 01010B and 10101B written into the WE4~WE0 bits, the second is using the Watchdog Timer software clear instruction and the third is via a HALT instruction.

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

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



**Watchdog Timer**
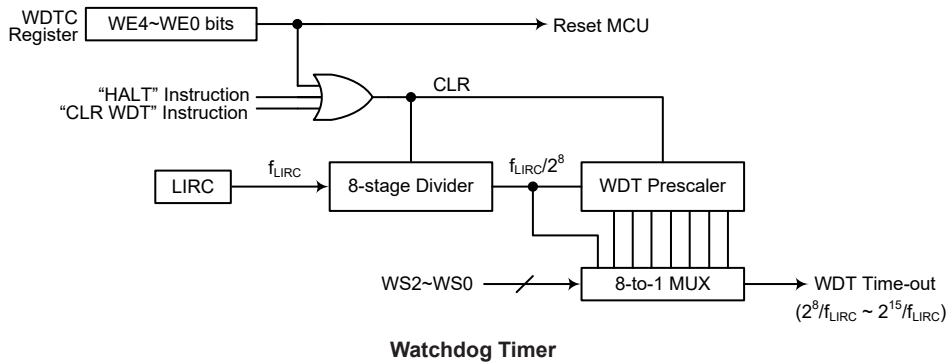
# Reset and Initialisation

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

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

## Reset Functions

There are several ways in which a microcontroller reset can occur, 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**
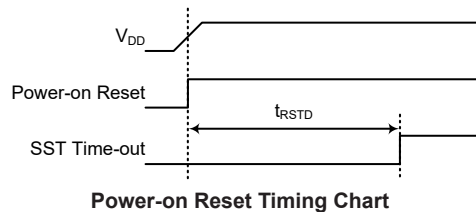
### Low Voltage Reset – LVR

The microcontroller contains a low voltage reset circuit in order to monitor the supply voltage of the device and provide an MCU reset when the value falls below a certain predefined level. If the supply voltage of the device drops to within a range of $0.9V \sim V_{LVR}$ 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 to 1. For a valid LVR signal, a low supply voltage, i.e., a voltage in the range between $0.9V \sim V_{LVR}$ must exist for a time greater than that specified by $t_{LVR}$ in the LVR Electrical Characteristics. If the low supply voltage state does not exceed this value, the LVR will ignore the low supply voltage and will not perform a reset function. 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 other 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.

Note that the LVR function will be automatically disabled when the device enters the SLEEP or IDLE mode.
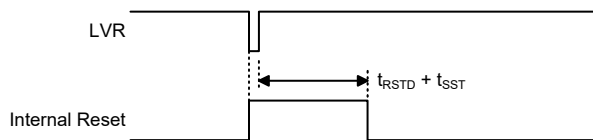
**Low Voltage Reset Timing Chart**

- **LVRC Register**

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

Bit 7~0        **LVS7~LVS0**: LVR voltage select
                     01011010: 2.1V
                     10100101: LVR disable
                     Other values: Generates a MCU reset – register is reset to POR value

                  When an actual low voltage condition occurs, as specified by the defined LVR voltage value above, an MCU reset will 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 values, 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 an actual 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 by the LVRC control register contains any undefined LVR voltage register values. This in effect acts like a software-reset function. Note that this bit can only be cleared to zero by the application program.

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

**Watchdog Time-out Reset during Normal Operation**

The Watchdog Time-out Reset during normal operation in the FAST or SLOW mode is the same as a Power On reset except that the Watchdog time-out flag TO will be set to "1".
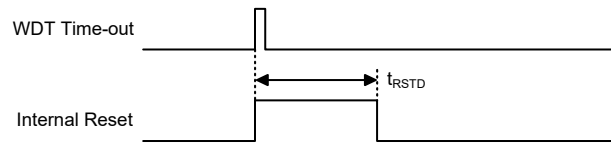


**WDT Time-out Reset during Normal Operation Timing Chart**

**Watchdog Time-out Reset during SLEEP or IDLE Mode**

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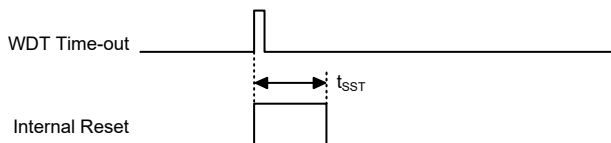


**WDT Time-out Reset during SLEEP or IDLE Timing Chart**

## Reset Initial Conditions

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

| TO | PDF | Reset Conditions |
|----|-----|------------------|
| 0 | 0 | Power-on reset |
| u | u | LVR reset during FAST or SLOW Mode operation |
| 1 | u | WDT time-out reset during FAST or SLOW Mode operation |
| 1 | 1 | WDT time-out reset during IDLE or SLEEP Mode operation |

"u" stands for unchanged

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

| Item | Condition after Reset |
|------|-----------------------|
| Program Counter | Reset to zero |
| Interrupts | All interrupts will be disabled |
| WDT, Time Base | Cleared after reset, WDT begins counting |
| Timer/Event Counter | Timer/Event Counter will be turned off |
| Input/Output Ports | I/O ports will be 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 normal program execution after a reset occurs, it is important to know what condition the microcontroller is in after a particular reset occurs. The following table describes how each type of reset affects each of the microcontroller internal registers. Note that where more than one package type exists the table will reflect the situation for the larger package type.

| Register | Power On Reset | WDT Time-out (Normal Operation) | WDT Time-out (IDLE/SLEEP) |
|---|---|---|---|
| IAR0 | x x x x  x x x x | u u u u  u u u u | u u u u  u u u u |
| MP0 | x x x x  x x x x | u u u u  u u u u | u u u u  u u u u |
| IAR1 | x x x x  x x x x | u u u u  u u u u | u u u u  u u u u |
| MP1 | x x x x  x x x x | u u u u  u u u u | u u u u  u u u u |
| BP | - - - -  - - - 0 | - - - -  - - - 0 | - - - -  - - - u |
| ACC | x x x x  x x x x | u u u u  u u u u | u u u u  u u u u |
| PCL | 0 0 0 0  0 0 0 0 | 0 0 0 0  0 0 0 0 | 0 0 0 0  0 0 0 0 |
| TBLP | x x x x  x x x x | u u u u  u u u u | u u u u  u u u u |
| TBLH | - x x x  x x x x | - u u u  u u u u | - u u u  u u u u |
| TBHP | - - - -  - x x x | - - - -  - u u u | - - - -  - u u u |
| STATUS | - - 0 0  x x x x | - - 1 u  u u u u | - - 1 1  u u u u |
| RSTFC | - - - -  - x 0 0 | - - - -  - u u u | - - - -  - u u u |
| SCC | 0 0 0 -  - - 0 0 | 0 0 0 -  - - 0 0 | u u u -  - - u u |
| HIRCC | - - - -  - - 0 1 | - - - -  - - 0 1 | - - - -  - - u u |
| WDTC | 0 1 0 1  0 0 1 1 | 0 1 0 1  0 0 1 1 | u u u u  u u u u |
| PSCR | - - - -  - 0 0 0 | - - - -  - 0 0 0 | - - - -  - u u u |
| PA | 1 1 1 1  1 1 1 1 | 1 1 1 1  1 1 1 1 | u u u u  u u u u |
| PAC | 1 1 1 1  1 1 1 1 | 1 1 1 1  1 1 1 1 | u u u u  u u u u |
| PAPU | 0 0 0 0  0 0 0 0 | 0 0 0 0  0 0 0 0 | u u u u  u u u u |
| PAWU | 0 0 0 0  0 0 0 0 | 0 0 0 0  0 0 0 0 | u u u u  u u u u |
| PB | 1 1 1 1  1 1 1 1 | 1 1 1 1  1 1 1 1 | u u u u  u u u u |
| PBC | 1 1 1 1  1 1 1 1 | 1 1 1 1  1 1 1 1 | u u u u  u u u u |
| PBPU | 0 0 0 0  0 0 0 0 | 0 0 0 0  0 0 0 0 | u u u u  u u u u |
| PC | - - - -  - - 1 1 | - - - -  - - 1 1 | - - - -  - - u u |
| PCC | - - - -  - - 1 1 | - - - -  - - 1 1 | - - - -  - - u u |
| PCPU | - - - -  - - 0 0 | - - - -  - - 0 0 | - - - -  - - u u |
| EEA | - - - 0  0 0 0 0 | - - - 0  0 0 0 0 | - - - u  u u u u |
| EED | 0 0 0 0  0 0 0 0 | 0 0 0 0  0 0 0 0 | u u u u  u u u u |
| TMR | 0 0 0 0  0 0 0 0 | 0 0 0 0  0 0 0 0 | u u u u  u u u u |
| TMRC | 0 0 0 0  1 0 0 0 | 0 0 0 0  1 0 0 0 | u u u u  u u u u |
| TBC | 0 - - -  - 0 0 0 | 0 - - -  - 0 0 0 | u - - -  - u u u |
| INTC0 | - 0 0 0  0 0 0 0 | - 0 0 0  0 0 0 0 | - u u u  u u u u |
| INTC1 | 0 0 0 0  0 0 0 0 | 0 0 0 0  0 0 0 0 | u u u u  u u u u |
| INTEG | - - - -  0 0 0 0 | - - - -  0 0 0 0 | - - - -  u u u u |
| PAS0 | 0 0 - -  0 0 - - | 0 0 - -  0 0 - - | u u - -  u u - - |
| PAS1 | - - 0 0  0 0 0 0 | - - 0 0  0 0 0 0 | - - u u  u u u u |
| PBS0 | 0 0 0 0  0 0 0 0 | 0 0 0 0  0 0 0 0 | u u u u  u u u u |
| PBS1 | 0 0 0 0  0 0 0 0 | 0 0 0 0  0 0 0 0 | u u u u  u u u u |
| SADOL | x x x x  - - - - | x x x x  - - - - | u u u u  - - - - (ADRFS=0) <br> u u u u  u u u u (ADRFS=1) |
| SADOH | x x x x  x x x x | x x x x  x x x x | u u u u  u u u u (ADRFS=0) <br> - - - -  u u u u (ADRFS=1) |
| SADC0 | 0 0 0 0  0 0 0 0 | 0 0 0 0  0 0 0 0 | u u u u  u u u u |
| SADC1 | 0 0 0 0  0 0 0 0 | 0 0 0 0  0 0 0 0 | u u u u  u u u u |
| SWS0 | - - - -  - 0 0 0 | - - - -  - 0 0 0 | - - - -  - u u u |

| Register | Power On Reset | WDT Time-out (Normal Operation) | WDT Time-out (IDLE/SLEEP) |
|---|---|---|---|
| LCS0 | 1000 0000 | 1000 0000 | uuuu uuuu |
| LCS1 | - - - - - - x x | - - - - - - x x | - - - - - - x x |
| OPC0 | - - - - 0000 | - - - - 0000 | - - - - uuuu |
| OPC1 | 0 - - 0 0000 | 0 - - 0 0000 | u - - u uuuu |
| OPOCAL | 0010 0000 | 0010 0000 | uuuu uuuu |
| SLEDC0 | 0000 0000 | 0000 0000 | uuuu uuuu |
| SLEDC1 | - - - - - - 0 0 | - - - - - - 0 0 | - - - - - - u u |
| SLEDC2 | - - - - - - 0 0 | - - - - - - 0 0 | - - - - - - u u |
| PWM0P | 0000 0000 | 0000 0000 | uuuu uuuu |
| PWM0D | 0000 0000 | 0000 0000 | uuuu uuuu |
| PWM0C | 0000 00 - - | 0000 00 - - | uuuu uu - - |
| PWM1P | 0000 0000 | 0000 0000 | uuuu uuuu |
| PWM1D | 0000 0000 | 0000 0000 | uuuu uuuu |
| PWM1C | 0000 00 - - | 0000 00 - - | uuuu uu - - |
| LVRC | 0101 1010 | 0101 1010 | uuuu uuuu |
| OTDC | 0 - - - - - - 0 | 0 - - - - - - 0 | u - - - - - - u |
| EEC | 0 - - - 0000 | 0 - - - 0000 | u - - - uuuu |

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

## Input/Output Ports

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

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

| Register Name | Bit | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| PA | PA7 | PA6 | PA5 | PA4 | PA3 | PA2 | PA1 | PA0 |
| PAC | PAC7 | PAC6 | PAC5 | PAC4 | PAC3 | PAC2 | PAC1 | PAC0 |
| PAPU | PAPU7 | PAPU6 | PAPU5 | PAPU4 | PAPU3 | PAPU2 | PAPU1 | PAPU0 |
| PAWU | PAWU7 | PAWU6 | PAWU5 | PAWU4 | PAWU3 | PAWU2 | PAWU1 | PAWU0 |
| PB | PB7 | PB6 | PB5 | PB4 | PB3 | PB2 | PB1 | PB0 |
| PBC | PBC7 | PBC6 | PBC5 | PBC4 | PBC3 | PBC2 | PBC1 | PBC0 |
| PBPU | PBPU7 | PBPU6 | PBPU5 | PBPU4 | PBPU3 | PBPU2 | PBPU1 | PBPU0 |
| PC | — | — | — | — | — | — | PC1 | PC0 |
| PCC | — | — | — | — | — | — | PCC1 | PCC0 |
| PCPU | — | — | — | — | — | — | PCPU1 | PCPU0 |

"—": Unimplemented, read as "0"

**I/O Logic Function Register List**

## Pull-high Resistors

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

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

- **PxPU Register**

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

**PxPUn**: I/O Port x Pin pull-high function control

    0: Disable

    1: Enable

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

## Port A Wake-up

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

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

- **PAWU Register**

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

Bit 7~0    **PAWU7~PAWU0**: PA7~PA0 wake-up function control

    0: Disable

    1: Enable

## I/O Port Control Registers

Each I/O port has its own control register known as PAC~PCC, to control the input/output configuration. With this control register, each CMOS output or input can be reconfigured dynamically under software control. Each pin of the I/O ports is directly mapped to a bit in its associated port control register. For the I/O pin to function as an input, the corresponding bit of the control register must be written as a "1". This will then allow the logic state of the input pin to be directly read by instructions. When the corresponding bit of the control register is written as a "0", the I/O pin will be 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, B and C. However, the actual available bits for each I/O Port may be different.

## I/O Port Source Current Control

Each pin in this device can be configured with different output source current which is selected by the corresponding source current selection bits. These source current selection bits are available when the corresponding pin is configured as a CMOS output. Otherwise, these select bits have no effect. Users should refer to the Input/Output Characteristics section to obtain the exact value for different applications.

| Register Name | Bit | | | | | | | |
|------|------|------|------|------|------|------|------|------|
| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| SLEDC0 | SLEDC07 | SLEDC06 | SLEDC05 | SLEDC04 | SLEDC03 | SLEDC02 | SLEDC01 | SLEDC00 |
| SLEDC1 | — | — | — | — | — | — | SLEDC11 | SLEDC10 |
| SLEDC2 | — | — | — | — | — | — | SLEDC21 | SLEDC20 |

**Source Current Selection Register List**

• **SLEDC0 Register**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|------|------|------|------|------|------|------|------|------|
| Name | SLEDC07 | SLEDC06 | SLEDC05 | SLEDC04 | SLEDC03 | SLEDC02 | SLEDC01 | SLEDC00 |
| 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    **SLEDC07~SLEDC06**: PB7~PB4 source current selection
    00: Source current = Level 0 (Min.)
    01: Source current = Level 1
    10: Source current = Level 2
    11: Source current = Level 3 (Max.)

Bit 5~4    **SLEDC05~SLEDC04**: PB3, PB1, PB0 source current selection
    00: Source current = Level 0 (Min.)
    01: Source current = Level 1
    10: Source current = Level 2
    11: Source current = Level 3 (Max.)

Bit 3~2    **SLEDC03~SLEDC02**: PA7~PA4 source current selection
    00: Source current = Level 0 (Min.)
    01: Source current = Level 1
    10: Source current = Level 2
    11: Source current = Level 3 (Max.)

Bit 1~0      **SLEDC01~SLEDC00**: PA3~PA0 source current selection
         00: Source current = Level 0 (Min.)
         01: Source current = Level 1
         10: Source current = Level 2
         11: Source current = Level 3 (Max.)

- **SLEDC1 Register**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | — | — | — | — | — | — | SLEDC11 | SLEDC10 |
| R/W | — | — | — | — | — | — | R/W | R/W |
| POR | — | — | — | — | — | — | 0 | 0 |

Bit 7~2      Unimplemented, read as "0"

Bit 1~0      **SLEDC11~SLEDC10**: PC1~PC0 source current selection
         00: Source current = Level 0 (Min.)
         01: Source current = Level 1
         10: Source current = Level 2
         11: Source current = Level 3 (Max.)

- **SLEDC2 Register**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | — | — | — | — | — | — | SLEDC21 | SLEDC20 |
| R/W | — | — | — | — | — | — | R/W | R/W |
| POR | — | — | — | — | — | — | 0 | 0 |

Bit 7~2      Unimplemented, read as "0"

Bit 1~0      **SLEDC21~SLEDC20**: PB2 source current selection
         00: Source current = Level 0 (Min.)
         01: Source current = Level 1
         10: Source current = Level 2
         11: Source current = Level 3 (Max.)

## 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 "x" Output Function Selection register "n", labeled as PxSn, 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 INTn, 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 set as an input by setting the corresponding bit in the I/O port control register. To correctly deselect the pin-shared function, the peripheral function should first be disabled and then the corresponding pin-shared function control register can be modified to select other pin-shared functions.

| Register Name | Bit | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| PAS0 | PAS07 | PAS06 | — | — | PAS03 | PAS02 | — | — |
| PAS1 | — | — | PAS15 | PAS14 | PAS13 | PAS12 | PAS11 | PAS10 |
| PBS0 | PBS07 | PBS06 | PBS05 | PBS04 | PBS03 | PBS02 | PBS01 | PBS00 |
| PBS1 | PBS17 | PBS16 | PBS15 | PBS14 | PBS13 | PBS12 | PBS11 | PBS10 |

**Pin-shared Function Selection Register List**

• **PAS0 Register**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | PAS07 | PAS06 | — | — | PAS03 | PAS02 | — | — |
| R/W | R/W | R/W | — | — | R/W | R/W | — | — |
| POR | 0 | 0 | — | — | 0 | 0 | — | — |

Bit 7~6    **PAS07~PAS06**: PA3 Pin-shared function selection
　　　　　　00: PA3/INT1
　　　　　　01: AN1
　　　　　　10: PA3/INT1
　　　　　　11: PA3/INT1

Bit 5~4    Unimplemented, read as "0"

Bit 3~2    **PAS03~PAS02**: PA1 Pin-shared function selection
　　　　　　00: PA1/INT0
　　　　　　01: AN0
　　　　　　10: PA1/INT0
　　　　　　11: PA1/INT0

Bit 1~0    Unimplemented, read as "0"

• **PAS1 Register**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | — | — | PAS15 | PAS14 | PAS13 | PAS12 | 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    Unimplemented, read as "0"

Bit 5~4    **PAS15~PAS14**: PA6 Pin-shared function selection
　　　　　　00: PA6
　　　　　　01: AN3
　　　　　　10: PA6
　　　　　　11: PA6

Bit 3~2    **PAS13~PAS12**: PA5 Pin-shared function selection
　　　　　　00: PA5
　　　　　　01: AN2
　　　　　　10: PA5
　　　　　　11: PA5

Bit 1~0        **PAS11~PAS10**: PA4 Pin-shared function selection
   00: PA4
   01: VREF
   10: PA4
   11: PA4

• **PBS0 Register**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | PBS07 | PBS06 | PBS05 | PBS04 | PBS03 | PBS02 | PBS01 | PBS00 |
| 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        **PBS07~PBS06**: PB3 Pin-shared function selection
   00: PB3
   01: PWM1B
   10: PB3
   11: PB3

Bit 5~4        **PBS05~PBS04**: PB2 Pin-shared function selection
   00: PB2
   01: PWM1
   10: PB2
   11: PB2

Bit 3~2        **PBS03~PBS02**: PB1 Pin-shared function selection
   00: PB1
   01: PWM0B
   10: PB1
   11: PB1

Bit 1~0        **PBS01~PBS00**: PB0 Pin-shared function selection
   00: PB0
   01: PWM0
   10: PB0
   11: PB0

• **PBS1 Register**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | PBS17 | PBS16 | PBS15 | PBS14 | PBS13 | PBS12 | PBS11 | PBS10 |
| 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        **PBS17~PBS16**: PB7 Pin-shared function selection
   00: PB7
   01: OPARO
   10: PB7
   11: PB7

Bit 5~4        **PBS15~PBS14**: PB6 Pin-shared function selection
   00: PB6
   01: OPINN0
   10: OPINP
   11: PB6

Bit 3~2        **PBS13~PBS12**: PB5 Pin-shared function selection
   00: PB5
   01: OPINN1
   10: PB5
   11: PB5

Bit 1~0    **PBS11~PBS10**: PB4 Pin-shared function selection
        00: PB4
        01: OPINP
        10: OPINN0
        11: PB4

## I/O Pin Structures

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



**Logic Function Input/Output Structure**
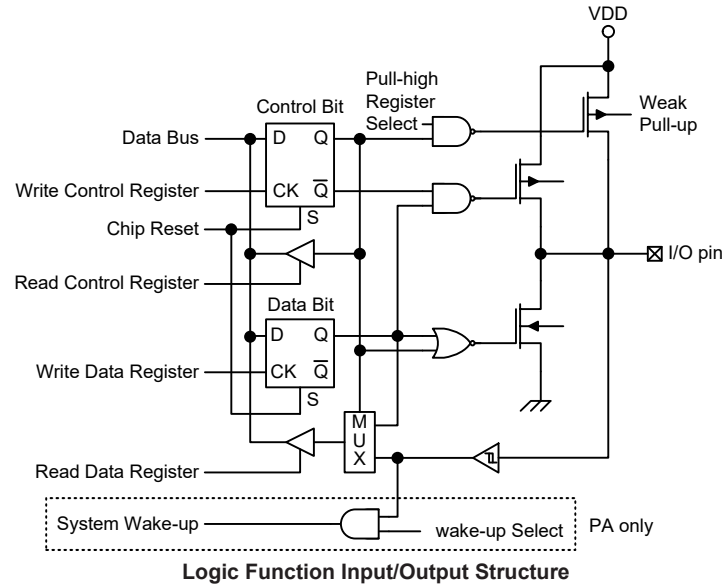
## Programming Considerations

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.

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

# Timer/Event Counter

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



**8-bit Timer/Event Counter**
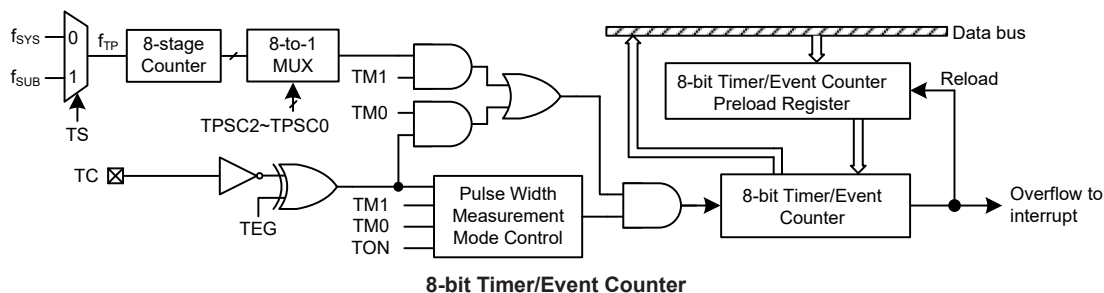
## Timer/Event Counter Input Clock Source

The Timer/Event Counter clock source can originate from various sources, an internal clock or an external pin. The internal clock source is used when the timer is in the Timer Mode and Pulse Width Measurement Mode. For the Timer/Event Counter, this internal clock source can be configured by the TS bit in the TMRC Timer Control Register to be derived from the $f_{SYS}$ or $f_{SUB}$ clock, the division ratio of which is selected by the TPSC2~TPSC0 bits in the TMRC Timer/Event Control Register.

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

## Timer/Event Counter Registers

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

| Register Name | Bit | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| TMRC | TM1 | TM0 | TS | TON | TEG | TPSC2 | TPSC1 | TPSC0 |
| TMR | D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 |

**Timer/Event Counter Register List**

### Timer Register – TMR

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

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

- **TMR Register**

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

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

### Timer Control Register – TMRC

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

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

To select which of the three modes the timer is to operate in, namely the Timer Mode, the Event Counter Mode or the Pulse Width Measurement Mode, the TM1~TM0 bits in the Timer Control Register must be set to the required logic levels. The timer-on bit TON provides the basic on/off control of the respective timer. Setting the bit to high allows the counter to run. Clearing the bit stops the counter. When the internal clock source is used, it can be sourced from the $f_{SYS}$ or $f_{SUB}$ clock selected by setting the TS bit. Bits TPSC2~TPSC0 determine the division ratio of the selected clock source. The internal clock selection will have no effect if an external clock source is used. If the timer is in the Event Counter or Pulse Width Measurement Mode, the active transition edge type is selected by the logic level of the TEG bit in the Timer Control Register.

- **TMRC Register**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|-----|-----|-----|-----|-----|-----|-----|-----|-----|
| Name | TM1 | TM0 | TS | TON | TEG | TPSC2 | TPSC1 | TPSC0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| POR | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |

Bit 7~6      **TM1~TM0**: Timer/Event Counter operating mode selection
          00: Unused
          01: Event Counter Mode
          10: Timer Mode
          11: Pulse Width Measurement Mode

Bit 5      **TS**: Timer $f_{TP}$ clock source selection
          0: $f_{SYS}$
          1: $f_{SUB}$

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

Bit 3      **TEG**: Timer/Event Counter active edge selection

Event Counter Mode
     0: Count on rising edge
     1: Count on falling edge

Pulse Width Measurement Mode
     0: Start counting on falling edge, stop on rising edge
     1: Start counting on rising edge, stop on falling edge

Bit 2~0      **TPSC2~TPSC0**: Timer internal clock selection
     000: $f_{TP}$
     001: $f_{TP}/2$
     010: $f_{TP}/4$
     011: $f_{TP}/8$
     100: $f_{TP}/16$
     101: $f_{TP}/32$
     110: $f_{TP}/64$
     111: $f_{TP}/128$

## Timer/Event Counter Operating Modes

The Timer/Event Counter can operate in one of three operating modes, Timer Mode, Event Counter Mode or Pulse Width Measurement Mode. The operating mode is selected using the TM1 and TM0 bits in the TMRC register.

### Timer Mode

To select this mode, bits TM1 and TM0 in the TMRC register should be set to "10" respectively. In this mode, the Timer/Event Counter can be utilised to measure fixed time intervals, providing an internal interrupt signal each time the Timer/Event Counter overflows.

When operating in this mode the internal clock $f_{TP}$ is used as the timer clock, which can be selected to be devirved from $f_{SYS}$ or $f_{SUB}$ by setting the TS bit in the TMRC register. The division of the $f_{TP}$ clock is selected by the TPSC2~TPSC0 bits in the same register. The timer-on bit TON must be set high to enable the timer to run. Each time an internal clock high to low transition occurs, the timer increases by one. When the timer reaches its maximum 8-bit, FFH Hex, value and overflows, an interrupt signal is generated and the timer will reload the value already loaded into the preload register and continue counting. It should be noted that in the Timer mode, even if the device is in the IDLE/SLEEP mode, if the selected internal clock is still activated and a timer overflow occurs, it will generate a timer interrupt and corresponding wake-up source.
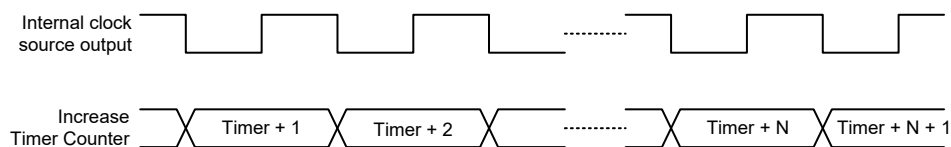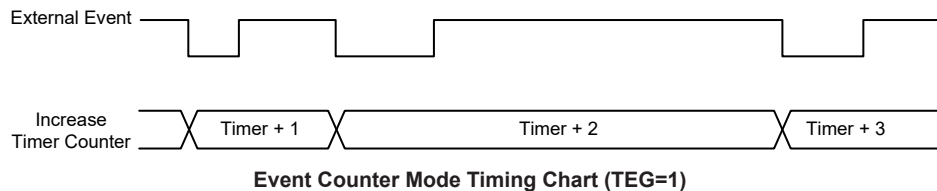


**Timer Mode Timing Chart**

### Event Counter Mode

To select this mode, bits TM1 and TM0 in the TMRC register should be set to "01" respectively. In this mode, a number of externally changing logic events, occurring on the external timer TC pin, can be recorded by the Timer/Event Counter.

When operating in this mode, the external timer pin, TC, is used as the Timer/Event Counter clock source. After the other bits in the Timer Control Register have been properly configured, the enable bit TON, can be set high to enable the Timer/Event Counter. If the Active Edge Selection bit, TEG, is low, the Timer/Event Counter will increase each time the TC pin receives a low to high

transition. If the TEG bit is high, the counter will increase each time the TC pin receives a high to low transition. When it is full and overflows, an interrupt signal is generated and the Timer/Event Counter will reload the value already loaded into the preload register and continue counting.

The external timer pin TC is pin-shared with PA7. The pin must be set as an input by setting the corresponding bit in the I/O port control register. It should be noted that in the Event Counter mode, even if the device is in the IDLE/SLEEP Mode, the Timer/Event Counter will continue to record externally changing logic events on the TC pin. As a result when the timer overflows it will generate a timer interrupt and corresponding wake-up source.



**Event Counter Mode Timing Chart (TEG=1)**

### Pulse Width Measurement Mode

To select this mode, bits TM1 and TM0 in the TMRC register should be set to "11" respectively. In this mode, the Timer/Event Counter can be utilised to measure the width of external pulses applied to the external timer pin.

When operating in this mode the internal clock $f_{TP}$ is used as the timer clock, which can be selected to be devirved from $f_{SYS}$ or $f_{SUB}$ by setting the TS bit in the TMRC register. The division of the $f_{TP}$ clock is selected by the TPSC2~TPSC0 bits in the same register. After the other bits in the Timer Control Register have been properly configured, the enable bit TON, can be set high to enable the Timer/Event Counter, however it will not actually start counting until an active edge is received on the TC pin.

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

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

The external timer pin TC is pin-shared with PA7. The pin must be set as an input by setting the corresponding bit in the I/O port control register. It should be noted that in the Pulse Width Measurement mode, even if the device is in the IDLE/SLEEP Mode, the Timer/Event Counter will continue to record externally changing logic events on the TC pin if the selected internal clock

source is still activated and the external signal continues to change state. As a result when the timer overflows it will generate a timer interrupt and corresponding wake-up source.



Internal Clock Output is sampled at every falling edge of T1.
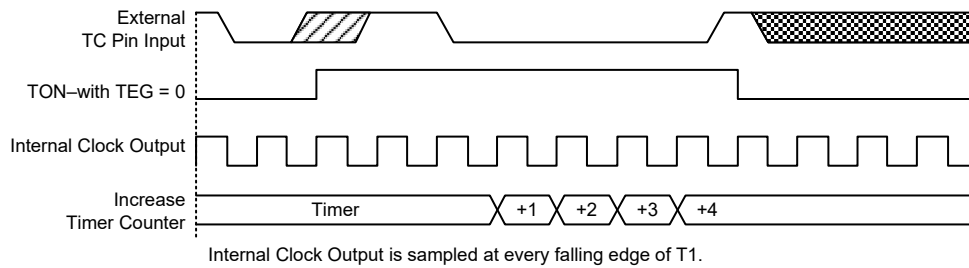
**Pulse Width Measurement Mode Timing Chart (TEG=0)**

## Programming Considerations

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

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

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

## Pulse Width Modulators

The device contains two 8-bit pulse width modulation functions. The PWM functions have complimentary outputs for maximum application flexibility.
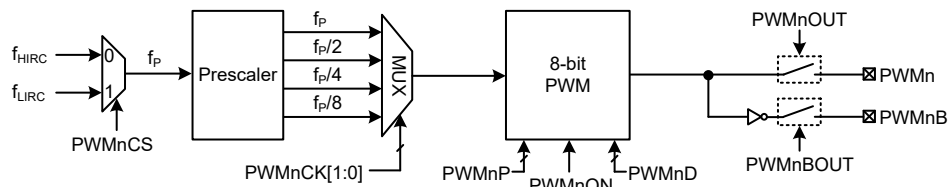


**PWM Block Diagram (n=0~1)**

### PWM Registers Description

There are three registers control the overall operation of each Pulse Width Modulator function. These are the PWM period register, PWMnP, PWM duty register, PWMnD and a single control register, PWMnC.

| Register Name | Bit | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| PWMnC | PWMnCK1 | PWMnCK0 | PWMnCS | PWMnON | PWMnOUT | PWMnBOUT | — | — |
| PWMnD | D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 |
| PWMnP | D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 |

**PWM Register List (n=0~1)**

- **PWMnC Register**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | PWMnCK1 | PWMnCK0 | PWMnCS | PWMnON | PWMnOUT | PWMnBOUT | — | — |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | — | — |
| POR | 0 | 0 | 0 | 0 | 0 | 0 | — | — |

Bit 7~6     **PWMnCK1~PWMnCK0**: PWMn counter clock selection
      00: $f_P$
      01: $f_P/2$
      10: $f_P/4$
      11: $f_P/8$

Bit 5     **PWMnCS**: PWMn counter clock source selection
      0: $f_{HIRC}$
      1: $f_{LIRC}$

Bit 4     **PWMnON**: PWMn function enable control
      0: Disable – PWMn counter=0
      1: Enable
    When the PWMnON bit is cleared to zero to disable the PWMn function, the external PWMn pin status will be floating when the multi-functional pins are selected as PWMn and PWMnB outputs.

Bit 3     **PWMnOUT**: PWMn output enable control
      0: Disable
      1: Enable
    When the PWMnOUT bit is cleared to zero, the external PWMn pin status will be floating when the multi-functional pin is selected as a PWMn output.

Bit 2            **PWMnBOUT**: PWMnB output enable control

   0: Disable

   1: Enable

When the PWMnBOUT bit is cleared to zero, the external PWMnB pin status will be floating when the multi-functional pin is selected as a PWMnB output.

Bit 1~0          Unimplemented, read as "0"

- **PWMnP 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**: 8-bit PWMn period register

- **PWMnD 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**: 8-bit PWMn duty register

If the PWMnD value is larger than or equal to the PWMnP value, the PWMn always outputs a high level. If the PWMnD value is equal to zero, the PWMn always outputs a low level.

## PWM Operation

The PWMn function is controlled by the PWMnON bit in the PWMnC register and its complementary outputs are controlled by the PWMnOUT and PWMnBOUT bits respectively. The PWMn signal generator is driven by the $f_{HIRC}$ or $f_{LIRC}$ clock selected by the PWMnCS bit, and can generate PWMn signals, with a variable duty and period cycles by configuring the 8-bit PWMnD and PWMnP registers respectively. The PWMn signal period is dependent upon the PWMn counter clock which is set by the PWMnCK1~PWMnCK0 bits in the PWMnC register and determined by the PWMnP register. The PWMn signal duty is determined by the PWMnD register content.

After the PWMnD and PWMnP register values are changed by the software, the new data will be updated by the hardware when the PWMn counter is cleared to zero.



**8-bit PWM Waveform**
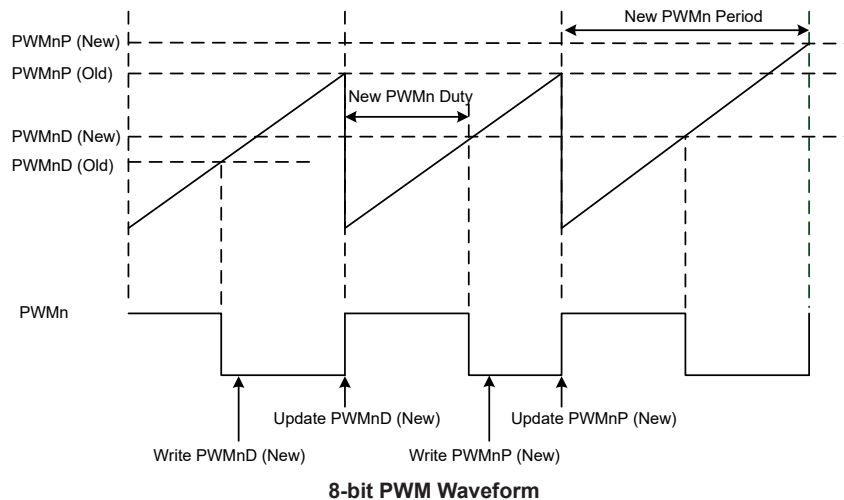
# Operational Amplifier

This device includes an operational amplifier for measure applications. An operational amplifier, OPAMP, produces an output potential that is typically hundreds of thousands of times larger than the potential difference between its input terminals. By integrating the OPAMP electronic circuitry into the microcontroller, the need for external components is reduced greatly.

## Operational Amplifier Operation

The Operational Amplifier can be used for signal amplification according to specific user requirements. The gain is selectable by using the OPG bit field together with the OPG2XEN bit. The amplified output can be directly output on the OPARO pin, and also be internally connected to the A/D converter for the amplified input signal read.
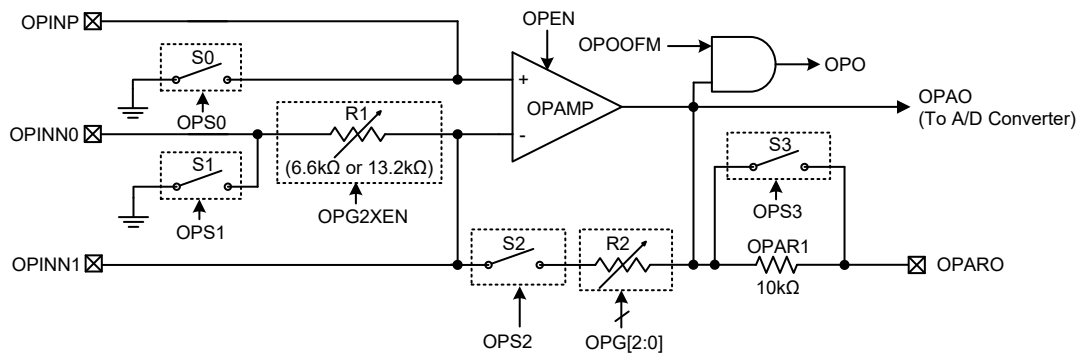


**Operational Amplifier Block Diagram**

## Operational Amplifier Registers

The internal Operational Amplifier normal operation and input offset voltage calibration function are controlled by three registers. The OPC0 register is used to control the switches on or off. The OPC1 register is used to control the OPAMP function on or off and select the gain. The OPO bit together with the OPOCAL register are used in the offset calibration procedure.

| Register Name | Bit | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| OPC0 | — | — | — | — | OPS3 | OPS2 | OPS1 | OPS0 |
| OPC1 | OPEN | — | — | OPO | OPG2XEN | OPG2 | OPG1 | OPG0 |
| OPOCAL | OPOOFM | OPORSP | OPOOF5 | OPOOF4 | OPOOF3 | OPOOF2 | OPOOF1 | OPOOF0 |

**Opetional Amplifier Register List**

### • OPC0 Register

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | — | — | — | — | OPS3 | OPS2 | OPS1 | OPS0 |
| R/W | — | — | — | — | R/W | R/W | R/W | R/W |
| POR | — | — | — | — | 0 | 0 | 0 | 0 |

Bit 7~4     Unimplemented, read as "0"

Bit 3       **OPS3**: Switch S3 on/off control
             0: Off
             1: On

Bit 2  **OPS2**: Switch S2 on/off control
     0: Off
     1: On

Bit 1  **OPS1**: Switch S1 on/off control
     0: Off
     1: On

Bit 0  **OPS0**: Switch S0 on/off control
     0: Off
     1: On

• **OPC1 Register**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|-----|---|---|---|---|---|---|---|---|
| Name | OPEN | — | — | OPO | OPG2XEN | OPG2 | OPG1 | OPG0 |
| R/W | R/W | — | — | R | R/W | R/W | R/W | R/W |
| POR | 0 | — | — | 0 | 0 | 0 | 0 | 0 |

Bit 7  **OPEN**: OPAMP function enable or disable control
     0: Disable
     1: Enable

Bit 6~5  Unimplemented, read as "0"

Bit 4  **OPO**: OPAMP output status (positive logic)
    When the OPOOFM bit is set to 1, OPO is defined as OPAMP output status, refer to Offset Calibration Procedure section. This bit is read only.

Bit 3  **OPG2XEN**: R2/R1 ratio doubling enable control
     0: Disable (Select R1=13.2kΩ)
     1: Enable (Select R1=6.6kΩ)
    When this bit is set to 1, the R2/R1 ratio selected by the OPG2~OPG0 bits will be doubled.

Bit 2~0  **OPG2~OPG0**: R2/R1 ratio selection
    If OPG2XEN=0, R1=13.2kΩ:
     000: R2/R1=2
     001: R2/R1=5
     010: R2/R1=6
     011: R2/R1=7
     100: R2/R1=20
     101: R2/R1=25
     110: R2/R1=30
     111: R2/R1=35
    If OPG2XEN=1, R1=6.6kΩ: The R2/R1 ratio doubling function is enabled, the above R2/R1 value will be doubled.

    Note that the internal resistors, R1 and R2, should be used when the gain is determined by these bits. This means the S1 switch should be on or OPINN should be selected together with the S2 switch on. Otherwise, the gain accuracy will not be guaranteed.

• **OPOCAL Register**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|-----|---|---|---|---|---|---|---|---|
| Name | OPOOFM | OPORSP | OPOOF5 | OPOOF4 | OPOOF3 | OPOOF2 | OPOOF1 | OPOOF0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| POR | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |

Bit 7  **OPOOFM**: OPAMP normal operation or input offset voltage calibration mode selection
     0: Normal operation
     1: Offset calibration mode

Bit 6      **OPORSP**: OPAMP input offset voltage calibration reference selection

       0: Select inverting input as the reference voltage input

       1: Select non-inverting input as the reference voltage input

Bit 5~0      **OPOOF5~OPOOF0**: OPAMP input offset voltage calibration control

This 6-bit field is used to perform the OPAMP input offset calibration operation and the value for the OPAMP input offset calibration can be restored into this bit field. More detailed information is described in the "Offset Calibration Procedure" section.

## Input voltage range

Together with different PGA operating modes, the input voltage on the OPAMP pins can be positive or negative for flexible application. There are two operating mode examples as the following. In these examples the internal resistors, R1 and R2, are used respectively.

- For $V_{IN} > 0$, the PGA operates in the non-inverting mode and the output voltage of the PGA is:
$$V_{OPGA}=(1+R2/R1)\times V_{IN}$$

- When the S0 and S2 switches are ON, S1 and S3 switches are OFF and the input node is OPINN. For $0 > V_{IN} > -0.2V$, the PGA operates in the inverting mode, the output voltage of the PGA is:
$$V_{OPGA}=-(R2/R1)\times V_{IN}$$

Note: If $V_{IN}$ is negative, it cannot be lower than -0.2V which will result in current leakage.

## Offset Calibration Function

This OPAMP includes an input offset calibration function. The calibrated data is stored in the OPOOF[5:0] bits. OPOOFM is calibration mode control bit and OPORSP is used to indicate that the input reference voltage comes from OPINP or OPINN1 in calibration mode. OPINP and OPINN1 are a input pair of OPAMP and the OPARO pin is the OPAMP analog output pin. The OPAMP digital output flag is OPO, which is used for OPAMP calibration mode.

### Offset Calibration Procedure

Note that as the OPAMP input pins are pin-shared with other functions, they should be configured as OPAMP inputs first by properly configuring the related pin-shared control bits. For operational amplifier input offset calibration, the procedures are summarized as the following.

Step1. Set OPOOFM=1 and OPORSP=1, the OPAMP is now under offset calibration mode. To make sure $V_{OS}$ as minimal as possible after calibration, the input reference voltage in calibration should be the same as input DC operating voltage in normal mode operation.

Step2. Set OPOOF[5:0]=000000 and then read OPO bit

Step3. Increase the OPOOF[5:0] value by 1 and then read the OPO bit.

If the OPO bit state has not changed, then repeat Step 3 until the OPO bit state has changed.

If the OPO bit state has changed, record the OPOOF[5:0] value as $V_{OS1}$ and then go to Step 4.

Step4. Set OPOOF[5:0]=111111 then read OPO bit.

Step5. Decrease the OPOOF[5:0] value by 1 and then read the OPO bit.

If the OPO bit state has not changed, then repeat Step 5 until the OPO bit state has changed.

If the OPO bit state has changed, record the OPOOF[5:0] value as $V_{OS2}$ and then go to Step 6.

Step6. Restore the Operational Amplifier input offset calibration value $V_{OS}$ into the OPOOF[5:0] bit field. The offset Calibration procedure is now finished.

$V_{OS}=(V_{OS1}+V_{OS2})/2$. If $(V_{OS1}+V_{OS2})/2$ is not integral, discard the decimal.

Residue $V_{OS}=V_{OUT}-V_{IN}$

# Analog to Digital Converter

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

## A/D Converter Overview

The device contains a multi-channel analog to digital converter which can directly interface to external analog signals, such as that from sensors or other control signals and convert these signals directly into a 12-bit digital value. It also can convert the internal signals, such as the divided battery voltage from the linear charger or OPAMP output signal, into a 12-bit digital value. The external or internal analog signal to be converted is determined by the SAINS2~SAINS0 bits together with the ACS2~ACS0 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 Signals | Channel Select Bits |
|---|---|---|
| 4: AN0~AN3 | 4: $V_{BAT}/2$, OPAO, $V_{DD}/4$, $V_{SS}$ | SAINS2~SAINS0, ACS2~ACS0 |

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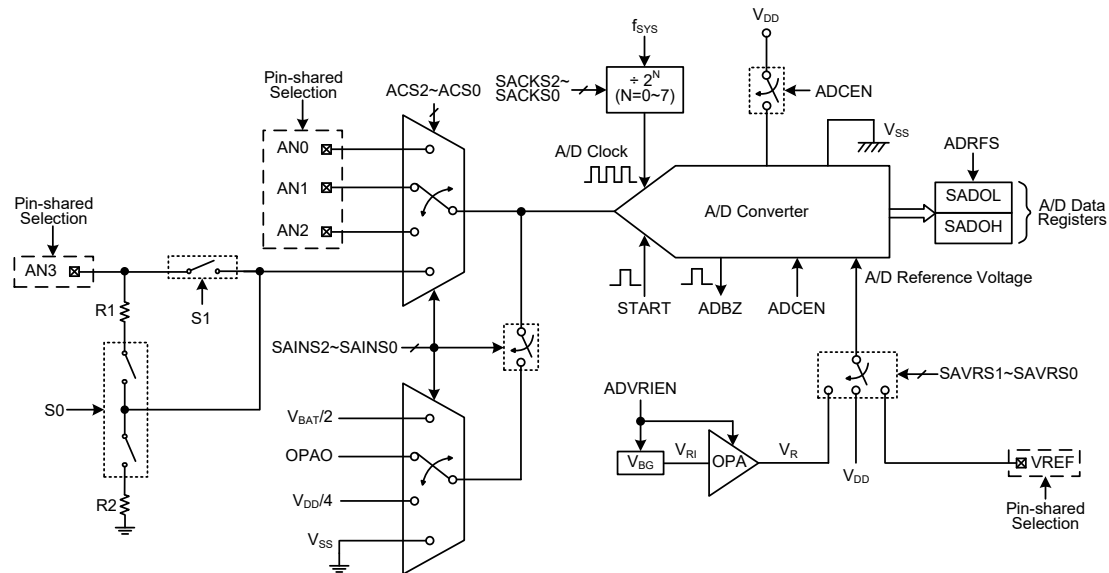


Note: $V_{BG}$ comes from the Linear Charger function.

**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 12-bit value. The remaining three registers are control registers which setup the operating and control function of the A/D converter.

| Register Name | Bit | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| SADOL (ADRFS=0) | D3 | D2 | D1 | D0 | — | — | — | — |
| SADOL (ADRFS=1) | D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 |
| SADOH (ADRFS=0) | D11 | D10 | D9 | D8 | D7 | D6 | D5 | D4 |
| SADOH (ADRFS=1) | — | — | — | — | D11 | D10 | D9 | D8 |
| SADC0 | START | ADBZ | ADCEN | ADRFS | ADVRIEN | ACS2 | ACS1 | ACS0 |
| SADC1 | SAINS2 | SAINS1 | SAINS0 | SAVRS1 | SAVRS0 | SACKS2 | SACKS1 | SACKS0 |
| SWS0 | — | — | — | — | — | D2 | S1 | S0 |

**A/D Converter Register List**

### A/D Converter Data Registers – SADOL, SADOH

As the device contains an internal 12-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 12 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~D11 are the A/D conversion result data bits. Any unused bits will be read as zero. Note that A/D Converter data register contents will be cleared 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 | D11 | D10 | D9 | D8 | D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 0 | 0 | D11 | D10 | D9 | D8 | D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 |

**A/D Data Registers**

### A/D Converter Control Registers – SADC0, SADC1, SWS0

To control the function and operation of the A/D converter, three control registers known as SADC0, SADC1 and SWS0 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, etc. As the device contains only one actual analog to digital converter hardware circuit, each of the external or internal analog signal inputs must be routed to the converter. The ACS2~ACS0 bits in the SADC0 register are used to determine which external channel input is selected to be converted. 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 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 | ADVRIEN | ACS2 | ACS1 | ACS0 |
| R/W | R/W | R | R/W | R/W | R/W | R/W | R/W | R/W |
| POR | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

Bit 7      **START**: Start the A/D conversion

         0→1→0: Start

This bit is used to initiate an A/D conversion process. The bit is normally low but if set high and then cleared low again, the A/D converter will initiate a conversion process.

Bit 6      **ADBZ**: A/D converter busy flag

         0: No A/D conversion is in progress
         1: A/D conversion is in progress

This read only flag is used to indicate whether the A/D conversion is in progress or not. When the START bit is set from low to high and then to low again, the ADBZ flag will be set to 1 to indicate that the A/D conversion is initiated. The ADBZ flag will be cleared to 0 after the A/D conversion is complete.

Bit 5      **ADCEN**: A/D converter function enable control

         0: Disable
         1: Enable

This bit controls the A/D internal function. This bit should be set to one to enable the A/D converter. If the bit is set low, then the A/D converter will be switched off reducing the device power consumption. When the A/D converter function is disabled, the contents of the A/D data register pair known as SADOH and SADOL will be cleared.

Bit 4      **ADRFS**: A/D converter data format selection

         0: A/D converter data format → SADOH=D[11:4]; SADOL=D[3:0]
         1: A/D converter data format → SADOH=D[11:8]; SADOL=D[7:0]

This bit controls the format of the 12-bit converted A/D value in the two A/D data registers. Details are provided in the A/D data register section.

Bit 3      **ADVRIEN**: A/D converter internal $V_{RI}$ and OPA (Gain=2) control

         0: Disable
         1: Enable

Bit 2~0    **ACS2~ACS0**: A/D converter external analog channel input selection

         000: AN0
         001: AN1
         010: AN2
         011: AN3
         100~111: Non-existed channel, the input will be floating

• **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: Internal input – Linear charger battery output voltage divided by 2, $V_{BAT}/2$
         010: Internal input – OPAMP output, OPAO
         011: Internal input – Ground, $V_{SS}$
         100: Internal input – A/D converter power supply voltage divided by 4, $V_{DD}/4$
         101~111: External input – External analog channel input

When the internal analog signal is selected to be converted, the external channel signal

input will automatically be switched off regardless of the ACS bit field value. It will prevent the external channel input from being connected together with the internal analog signal.

Bit 4~3   **SAVRS1~SAVRS0**: A/D converter reference voltage selection
  00: From external VREF pin
  01: From internal A/D converter power supply, $V_{DD}$
  10: From internal A/D converter OPA output voltage, $V_R$
  11: From external VREF pin

These bits are used to select the A/D converter reference voltage. When the internal reference voltage source is selected, the reference voltage derived from the external VREF pin will automatically be switched off.

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

These three bits are used to select the clock source for the A/D converter.

- **SWS0 Register**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|-----|---|---|---|---|---|---|---|---|
| Name | — | — | — | — | — | D2 | S1 | S0 |
| R/W | — | — | — | — | — | R/W | R/W | R/W |
| POR | — | — | — | — | — | 0 | 0 | 0 |

Bit 7~3   Unimplemented, read as "0"

Bit 2   **D2**: Reserved bit, must be fixed at "0"

Bit 1   **S1**: Internal voltage dividing resistor bypass control
  0: S1 disable
  1: S1 enable
  The S1 switch will be enabled only when the ACS bit field is set to select AN3 as the A/D input and S1=1, otherwise it will be disabled.

Bit 0   **S0**: Internal voltage dividing resistors R1 and R2 control
  0: S0 disable
  1: S0 enable
  The S0 switch will be enabled only when the ACS bit field is set to select AN3 as the A/D input and S1~S0 bits are set to 01, otherwise it will be disabled.

## 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 maximum 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 or larger than the minimum or maximum A/D clock period which may result in inaccurate A/D conversion values. Refer to the following table for examples, where values marked with an asterisk * show where special care must be taken, as the values may be less or larger than the specified A/D Clock Period range.

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

**A/D Clock Period Examples**

Controlling the power on/off function of the A/D converter circuitry is implemented using the ADCEN bit in the SADC0 register. This bit must be set high to power on the A/D converter. When the ADCEN bit is set high to power on the A/D converter internal circuitry, a certain delay, as indicated in the timing diagram, must be allowed before an A/D conversion is initiated. Even if no pins are selected for use as A/D inputs, 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 power supply $V_{DD}$, or from the A/D converter OPA output $V_R$, 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 $V_{DD}$. When the SAVRS bit filed is set to "10", the A/D converter reference voltage will come from the OPA output. Otherwise, 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 function. If the internal A/D converter power $V_{DD}$ or the OPA output $V_R$ is selected as the reference voltage, the external reference input from the VREF pin will automatically be switched off by hardware.

The analog input values must not be allowed to exceed the value of the selected A/D 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 | $V_R$ | Internal A/D converter OPA 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 PxS0 and PxS1 registers determine whether the input pins are setup as A/D converter analog input channel 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.

There are some internal analog signals derived from the linear charger battery outout, OPAMP output or A/D converter power supply, which can be connected to the A/D converter as the analog input signal by configuring the SAINS2~SAINS0 bits. If the external channel input is selected to be converted, the SAINS2~SAINS0 bits should be set to "000", "101", "110" or "111" and the ACS2~ACS0 bits can determine which external channel is selected. If the internal analog signal is selected to be converted, the external channel signal input will automatically be switched off regardless of the ACS bit field value. It will prevent the external channel input from being connected together with the internal analog signal.

| SAINS[2:0] | ACS[2:0] | Input Signals | Description |
|---|---|---|---|
| 000, 101~111 | 000~011 | AN0~AN3 | External pin analog input |
| | 100~111 | — | Non-existed channel, input is floating |
| 001 | xxx | $V_{BAT}/2$ | Linear charger battery output voltage divided by 2 |
| 010 | xxx | OPAO | OPAMP output voltage |
| 011 | xxx | $V_{SS}$ | Connected to ground |
| 100 | xxx | $V_{DD}/4$ | A/D converter power supply voltage divided by 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 12 A/D clock cycles. Therefore, a total of 16 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 ÷ 16

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 16 $t_{ADCK}$ clock cycles where $t_{ADCK}$ is equal to the A/D clock period.

**A/D Conversion Timing – External Channel Input**
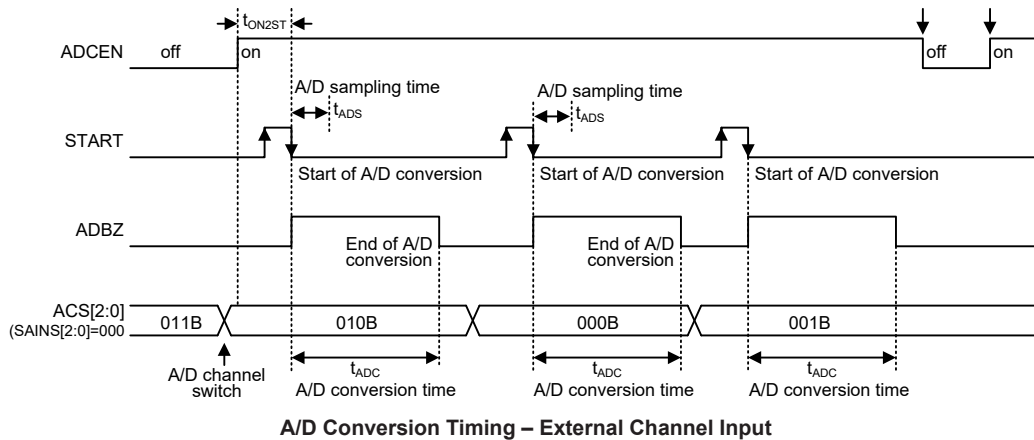
## Summary of A/D Conversion Steps

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 converter by setting the ADCEN bit in the SADC0 register to 1.

- Step 3

  Select which signal is to be connected to the internal A/D converter by correctly configuring the SAINS2~SAINS0 bits.

  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 bit field, the corresponding pins should be configured as A/D input function by configuring the relevant pin-shared function control bits. The desired analog channel then should be selected by configuring the ACS2~ACS0 bits. After this step, go to Step 6.

- Step 5

  If the SAINS bit field is set to "001"~"100", the relevant internal analog signal will be selected. When the internal analog signal is selected to be converted, the external channel analog input will automatically be disconnected. 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 internal reference voltage source is selected, the external reference voltage supplied on the VREF pin will be automatically disconnected. Set the ADVRIEN bit high if the OPA output voltage, $V_R$, is selected as the A/D converter reference voltage.

- Step 7

  Select A/D converter output data format by setting the ADRFS bit in the SADC0 register.

- Step 8

  If A/D conversion interrupt is used, the interrupt control registers must be correctly configured to ensure the A/D interrupt function is active. The master interrupt control bit, EMI, and the A/D

conversion interrupt control bit, ADE, must both be set high in advance.

- Step 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 normal I/O pins, 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 12-bit A/D converter, its full-scale converted digitised value is equal to FFFH. 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 4096.

$$1 \text{ LSB} = V_{REF} \div 4096$$

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

$$\text{A/D input voltage} = \text{A/D output digital value} \times (V_{REF} \div 4096)$$

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 bit field.
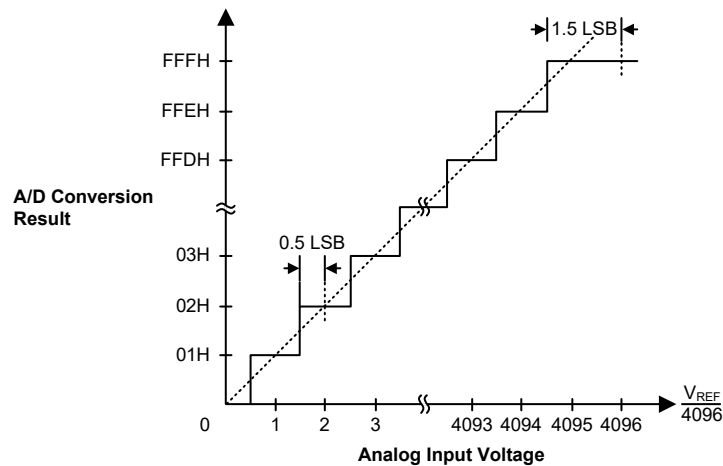


**Ideal A/D Transfer Function**

### 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, 0BH
mov  SADC1, a         ; select input signal from external channel input,
                      ; reference voltage from VDD fSYS/8 as A/D clock
mov  a, 04h           ; setup PAS0 register to configure pin AN0
mov  PAS0, a
mov  a, 20h
mov  SADC0, a         ; enable A/D converter and connect AN0 channel to A/D converter
:
start_conversion:
clr  START            ; high pulse on start bit to initiate conversion
set  START            ; reset A/D
clr  START            ; start A/D
polling_EOC:
sz   ADBZ             ; poll the SADC0 register ADBZ bit to detect end of A/D
                      ; conversion
jmp  polling_EOC      ; continue polling
mov  a, 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, 0BH
mov  SADC1, a         ; select input signal from external channel input,
                      ; reference voltage from AVDD fSYS/8 as A/D clock
mov  a, 04h           ; setup PAS0 register to configure pin AN0
mov  PAS0, a
mov  a, 20h
mov  SADC0, a         ; enable A/D converter and connect AN0 channel to A/D converter
Start_conversion:
clr  START            ; high pulse on START bit to initiate conversion
set  START            ; reset A/D
clr  START            ; start A/D
clr  ADF              ; clear ADC interrupt request flag
set  ADE              ; enable ADC interrupt
set  EMI              ; enable global interrupt
:
:
; ADC interrupt service routine
ADC_ISR:
mov  acc_stack, a     ; save ACC to user defined memory
mov  a, STATUS
mov  status_stack, a  ; save STATUS to user defined memory
:
```

```
                      :
mov  a, 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
```

## Linear Charger

This device includes a Linear Charger which can work in three modes.
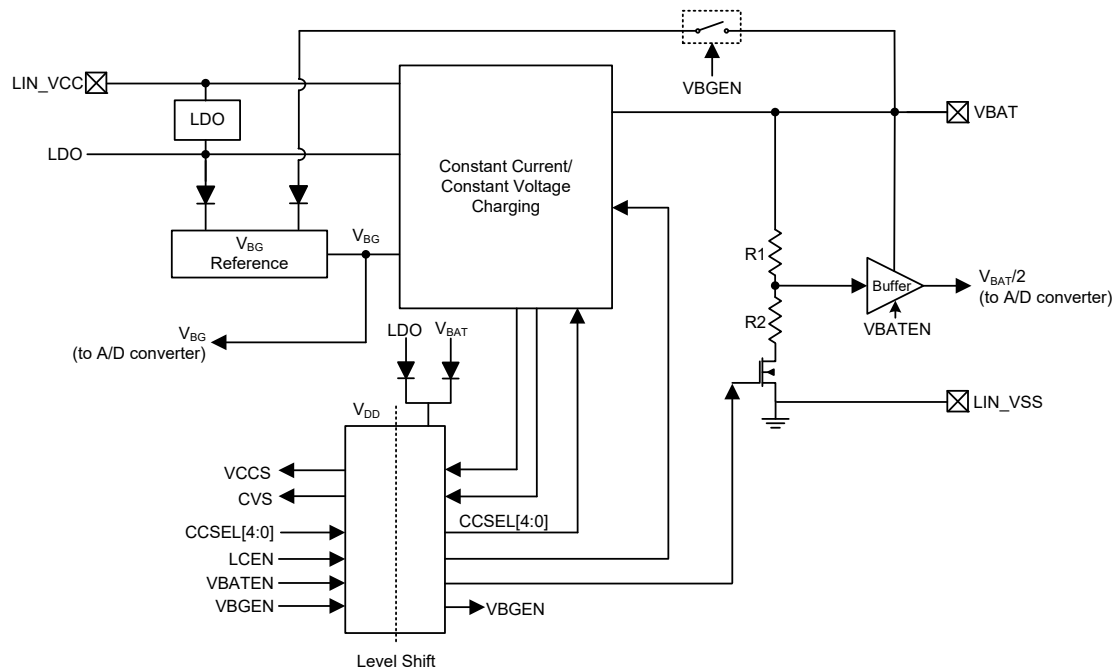


**Linear Charger Structure**

### Linear Charger Control Register

There are two registers for the linear charger overall control and monitoring.

| Register Name | Bit | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| LCS0 | LCEN | VBGEN | VBATEN | CCSEL4 | CCSEL3 | CCSEL2 | CCSEL1 | CCSEL0 |
| LCS1 | — | — | — | — | — | — | VCCS | CVS |

**Timer/Event Counter Register List**

• **LCS0 Register**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|-----|---|---|---|---|---|---|---|---|
| Name | LCEN | VBGEN | VBATEN | CCSEL4 | CCSEL3 | CCSEL2 | CCSEL1 | CCSEL0 |
| 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 |

Bit 7      **LCEN**: Charging function control
    0: Disable
    1: Enable
When $V_{BAT}$<2.2V, the LCEN bit setting is invalid, the charging function is always on.
When $V_{BAT}$>2.2V, the charging function on/off is controlled by the LCEN bit setting.

Bit 6      **VBGEN**: $V_{BG}$ reference voltage control
    0: Disable, $V_{BAT}$ does not supply power to $V_{BG}$
    1: Enable, $V_{BAT}$ supplies power to $V_{BG}$

Bit 5      **VBATEN**: Voltage dividing resistors and buffer control
    0: Voltage dividing resistors off, buffer off; buffer output=0
    1: Voltage dividing resistors on, buffer on; buffer output=$V_{BAT}$/2

Bit 4~0    **CCSEL4~CCSEL0**: Constant current $I_{BAT\_CC}$ selection
    00000: 200mA
    00001: 220mA
    :     : (+20mA/step)
    10110: 640mA
    10111: 680mA
    :     : (+40mA/step)
    11111: 1000mA
Before the MCU power on reset is finished, the bit setting is invalid and the $I_{BAT\_CC}$
value is fixed at 200mA.

• **LCS1 Register**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|-----|---|---|---|---|---|---|---|---|
| Name | — | — | — | — | — | — | VCCS | CVS |
| R/W | — | — | — | — | — | — | R | R |
| POR | — | — | — | — | — | — | x | x |

"x": unknown

Bit 7~2    Unimplemented, read as "0"

Bit 1      **VCCS**: Linear charger input voltage $V_{CC}$ status
    0: Input voltage $V_{CC}$ is abnormal
    1: Input voltage is within the charging range: ($V_{BAT}$+0.1V) < $V_{CC}$ < $V_{OVP}$

Bit 0      **CVS**: Battery fully charged flag
    0: The battery is not fully charged
    1: The battery has been fully charged
This flag is valid only when the VCCS flag is high.

## Functional Description

There are three charge control modes for the Linear Charger, which are the trickle mode, constant current charge mode and constant voltage charge mode. The required charge voltage and current for these modes differ according to different user designs. These charge modes are described below.

Trickle Mode: This mode is used for completely discharged batteries which have a battery voltage of less than 3V. When the battery voltage is less than 2.2V, the battery will be pre-charged using a typical constant current value of 20mA during this first charge stage. When the battery voltage is larger than 2.2V, the battery will be pre-charged using a typical current value of $0.1 \times I_{BAT\_CC}$ during this first charge stage.

Constant Current Charging Mode: When the battery voltage is larger than 3V and less than 4.2V, the charge current will then switch to a typical constant current $I_{BAT\_CC}$ during this second charge stage. The charge current is determined by the CCSEL4~CCSEL0 bits in the LCS0 register and is within the range of 200mA to 1000mA.

Constant Voltage Charging Mode: Once the battery voltage has reached 4.2V, it will be charged using a constant voltage during this third stage. The charge voltage should be fixed at 4.2V with a tolerence within 1%. The charge current gradually decreases as the constant voltage charge time increases. Typically, the constant voltage charge stage is completed when the charge current reduces to a value less than $0.1 \times I_{BAT\_CC}$. At this time the battery is fully charged and the CVS bit will be set high. After charging, the battery voltage will always be monitored. When the battery voltage is less than 4.0V, the battery will be recharged and the next charge cycle will be entered.
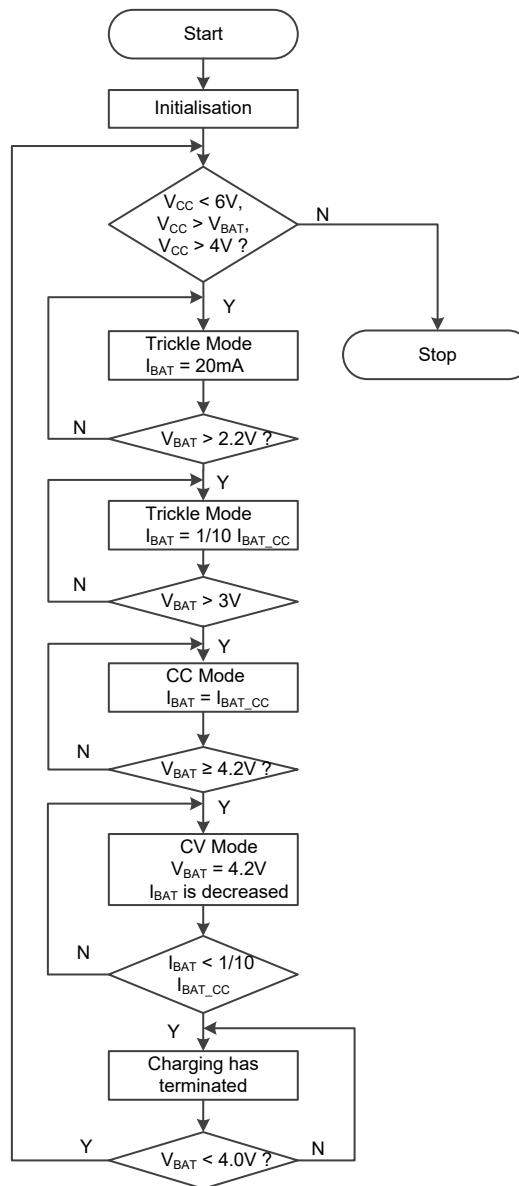


**Linear Charger Flowchart**

# Over Temperature Detector

The device includes an over temperature detector which is controlled by the OTDEN bit. When the over temperature detector has been enabled and the chip junction temperature exceeds 110°C, the temperature protection circuit will set the OTDF flag high. When the chip junction temperature drops below 85°C, the OTDF flag will be cleared to zero.

The over temperature detector has its own interrupt, providing an alternative means of over temperature detection, in addition to polling the OTDF flag. The interrupt will only be generated after a delay time of $t_{OTD}$ after OTDF is set high.
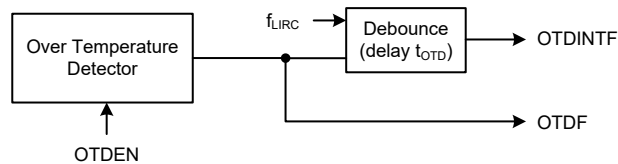


**Over Temperature Detector Diagram**

• **OTDC Register**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|------|-------|---|---|---|---|---|---|------|
| Name | OTDEN | — | — | — | — | — | — | OTDF |
| R/W  | R/W   | — | — | — | — | — | — | R |
| POR  | 0     | — | — | — | — | — | — | 0 |

Bit 7        **OTDEN**: Over temperature detection function control
             0: Disable
             1: Enable

Bit 6~1      Unimplemented, read as "0"

Bit 0        **OTDF**: Over temperature detection flag
             0: Temperature < 85°C
             1: Temperature > 110°C

# Interrupts

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

## 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. The registers fall into two categories. The first is the INTC0~INTC1 registers which setup the primary interrupts, the second is the INTEG register which sets 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 | — | — |
| INTn Pin | INTnE | INTnF | n=0~1 |
| Timer/Event Counter | TE | TF | — |
| Time Base | TBE | TBF | — |
| A/D Converter | ADE | ADF | — |
| EEPROM | DEE | DEF | — |
| Over Temperature Detector | OTDINTE | OTDINTF | — |

**Interrupt Register Bit Naming Conventions**

| Register Name | Bit | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| INTEG | — | — | — | — | INT1S1 | INT1S0 | INT0S1 | INT0S0 |
| INTC0 | — | TBF | TF | INT0F | TBE | TE | INT0E | EMI |
| INTC1 | OTDINTF | INT1F | DEF | ADF | OTDINTE | INT1E | DEE | ADE |

**Interrupt Register List**

• **INTEG Register**

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

Bit 7~4      Unimplemented, read as "0"

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

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

• **INTC0 Register**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | — | TBF | TF | INT0F | TBE | TE | INT0E | EMI |
| R/W | — | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| POR | — | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

Bit 7      Unimplemented, read as "0"

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

Bit 5      **TF**: Timer/Event Counter interrupt request flag
        0: No request
        1: Interrupt request

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

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

Bit 2       **TE**: Timer/Event Counter interrupt control
            0: Disable
            1: Enable

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

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

• **INTC1 Register**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | OTDINTF | INT1F | DEF | ADF | OTDINTE | INT1E | DEE | ADE |
| 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       **OTDINTF**: Over Temperature Detector interrupt request flag
            0: No request
            1: Interrupt request

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

Bit 5       **DEF**: Data EEPROM interrupt request flag
            0: No request
            1: Interrupt request

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

Bit 3       **OTDINTE**: Over Temperature Detector interrupt control
            0: Disable
            1: Enable

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

Bit 1       **DEE**: Data EEPROM interrupt control
            0: Disable
            1: Enable

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

## Interrupt Operation

When the conditions for an interrupt event occur, such as a timer overflow or 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 normal execution at the point where the interrupt occurred.

The various interrupt enable bits, together with their associated request flags, are shown in the accompanying diagrams with their order of priority. All 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. All of the interrupt request flags when set will wake up the device if it is in SLEEP or IDLE Mode, however to prevent a wake-up from occurring the corresponding flag should be set before the device are in SLEEP or IDLE Mode.



**Interrupt Structure**
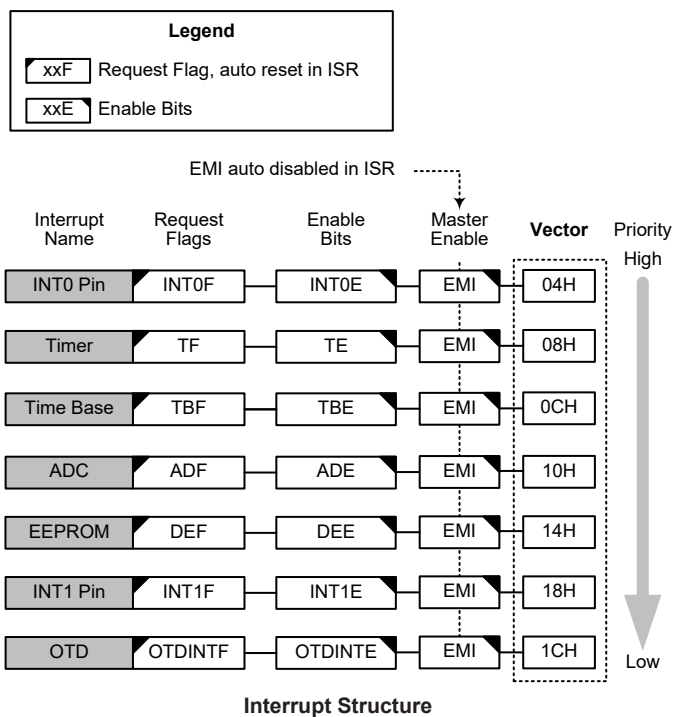
## External Interrupts

The external interrupt is controlled by signal transitions on the INTn pin. An external interrupt request will take place when the external interrupt request flag, INTnF, is set, which will occur when a transition, whose type is chosen by the edge selection bits, appears on the external interrupt pin. To allow the program to branch to its interrupt vector address, the global interrupt enable bit, EMI, and the external interrupt enable bit, INTnE, must first be set. Additionally, the correct interrupt edge type must be selected using the INTEG register to enable the external interrupt function and to choose the trigger edge type. As the external interrupt pin is pin-shared with I/O pin, it can only be configured as external interrupt pin if its external interrupt enable bit in the corresponding interrupt register has been set and the external interrupt pin is selected by the corresponding pin-shared function selection bits. The pin must also be 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, INTnF, 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.

## Timer/Event Counter Interrupt

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

## Time Base Interrupt

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

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

**Time Base Interrupt**
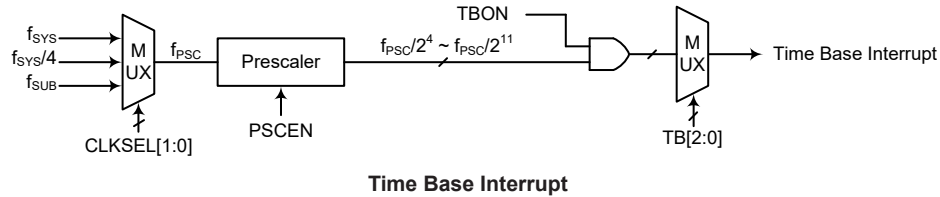
• **PSCR Register**

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

Bit 7~3    Unimplemented, read as "0"

Bit 2    **PSCEN**: Prescaler clock enable control
　　　　0: Disable
　　　　1: Enable

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

• **TBC Register**

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

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

Bit 6~3    Unimplemented, read as "0"

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

## A/D Converter Interrupt

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

### EEPROM Interrupt

An EEPROM Interrupt request will take place when the EEPROM Interrupt request flag, DEF, is set, which occurs when an EEPROM Write cycle ends. To allow the program to branch to its interrupt vector address, the global interrupt enable bit, EMI, and the EEPROM Interrupt enable bit, DEE, must first be set. When the interrupt is enabled, the stack is not full and an EEPROM Write cycle ends, a subroutine call to the EEPROM Interrupt vector will take place. When the EEPROM Interrupt is serviced, the DEF flag will be automatically cleared and the EMI bit will also be automatically cleared to disable other interrupts.

### Over Temperature Detector Interrupt

An Over Temperatuere Detector Interrupt request will take place when the Over Temperatuere Detector Interrupt request flag, OTDINTF, is set, which occurs when the chip junction temperature exceeds the over temperature threshold. To allow the program to branch to its interrupt vector address, the global interrupt enable bit, EMI, and the Over Temperatuere Detector Interrupt enable bit, OTDINTE, must first be set. When the interrupt is enabled, the stack is not full and the chip junction temperature exceeds the over temperature threshold, a subroutine call to the Over Temperatuere Detector Interrupt vector will take place. When the interrupt is serviced, the OTDINTF flag will be automatically cleared and the EMI bit will also be automatically cleared to disable other interrupts.

### Interrupt Wake-up Function

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

### Programming Considerations

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

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

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

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

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

## Application Circuits
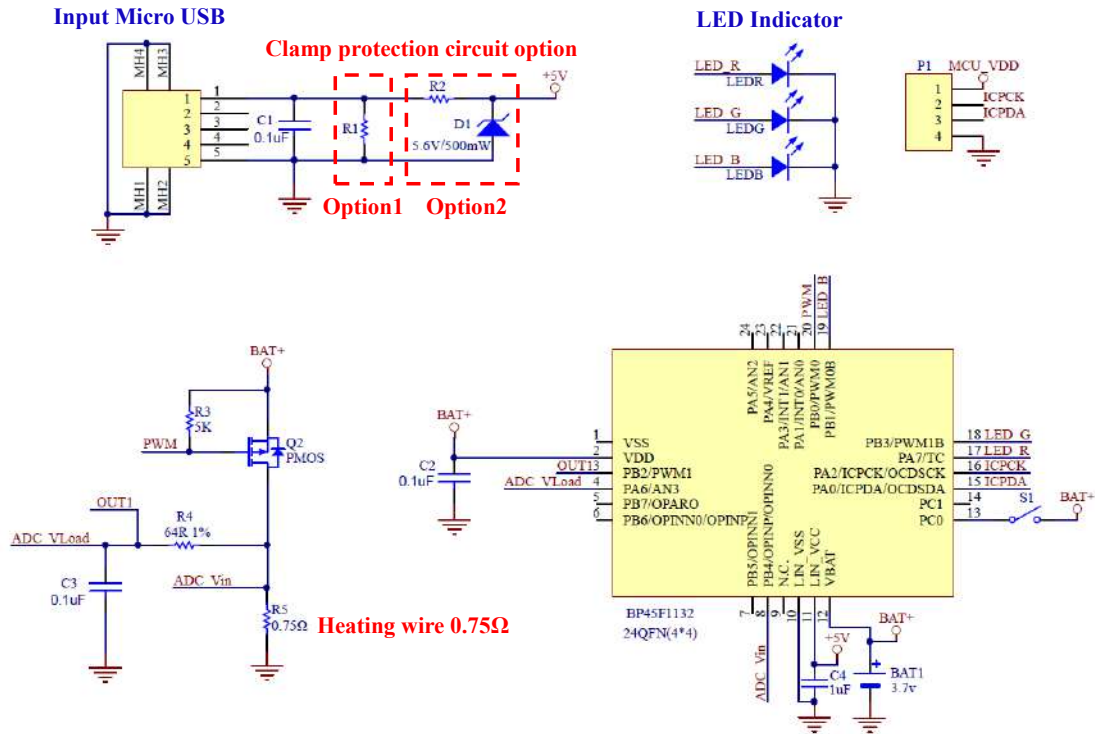
# Instruction Set

## Introduction

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

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

## Instruction Timing

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

## Moving and Transferring Data

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

## Arithmetic Operations

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

### Logical and Rotate Operation

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

### Branches and Control Transfer

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

### Bit Operations

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

### Table Read Operations

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

### Other Operations

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

## Instruction Set Summary

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

### Table Conventions

x: Bits immediate data

m: Data Memory address

A: Accumulator

i: 0~7 number of bits

addr: Program memory address

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

| Mnemonic | Description | Cycles | Flag Affected |
|----------|-------------|--------|---------------|
| **Data Move** | | | |
| MOV A,[m] | Move Data Memory to ACC | 1 | None |
| MOV [m],A | Move ACC to Data Memory | 1[Note] | None |
| MOV A,x | Move immediate data to ACC | 1 | None |
| **Bit Operation** | | | |
| CLR [m].i | Clear bit of Data Memory | 1[Note] | None |
| SET [m].i | Set bit of Data Memory | 1[Note] | None |
| **Branch Operation** | | | |
| JMP addr | Jump unconditionally | 2 | None |
| SZ [m] | Skip if Data Memory is zero | 1[Note] | None |
| SZA [m] | Skip if Data Memory is zero with data movement to ACC | 1[Note] | None |
| SZ [m].i | Skip if bit i of Data Memory is zero | 1[Note] | None |
| SNZ [m].i | Skip if bit i of Data Memory is not zero | 1[Note] | None |
| SIZ [m] | Skip if increment Data Memory is zero | 1[Note] | None |
| SDZ [m] | Skip if decrement Data Memory is zero | 1[Note] | None |
| SIZA [m] | Skip if increment Data Memory is zero with result in ACC | 1[Note] | None |
| SDZA [m] | Skip if decrement Data Memory is zero with result in ACC | 1[Note] | 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 | 2[Note] | None |
| TABRDL [m] | Read table (last page) to TBLH and Data Memory | 2[Note] | None |
| **Miscellaneous** | | | |
| NOP | No operation | 1 | None |
| CLR [m] | Clear Data Memory | 1[Note] | None |
| SET [m] | Set Data Memory | 1[Note] | None |
| CLR WDT | Clear Watchdog Timer | 1 | TO, PDF |
| SWAP [m] | Swap nibbles of Data Memory | 1[Note] | 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      $ACC \leftarrow ACC + [m] + C$

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      $[m] \leftarrow ACC + [m] + C$

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      $ACC \leftarrow ACC + [m]$

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      $ACC \leftarrow ACC + x$

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      $[m] \leftarrow ACC + [m]$

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      $ACC \leftarrow ACC \; ''AND'' \; [m]$

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      $ACC \leftarrow ACC \; ''AND'' \; x$

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      $[m] \leftarrow ACC \; ''AND'' \; [m]$

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 | $[m] \leftarrow [m] - 1$ |
| 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 | $ACC \leftarrow [m] - 1$ |
| 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 | $TO \leftarrow 0$<br>$PDF \leftarrow 1$ |
| Affected flag(s) | TO, PDF |

| | |
|---|---|
| **INC [m]** | Increment Data Memory |
| Description | Data in the specified Data Memory is incremented by 1. |
| Operation | $[m] \leftarrow [m] + 1$ |
| 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 | $ACC \leftarrow [m] + 1$ |
| 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 $\leftarrow$ 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 | $ACC \leftarrow [m]$ |
| Affected flag(s) | None |

| | |
|---|---|
| **MOV A,x** | Move immediate data to ACC |
| Description | The immediate data specified is loaded into the Accumulator. |
| Operation | $ACC \leftarrow x$ |
| 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 | $[m] \leftarrow ACC$ |
| 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<br>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<br>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)<br>[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)<br>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)<br>[m].0 ← C<br>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)<br>ACC.0 ← C<br>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)<br>[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)<br>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)<br>[m].7 ← C<br>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)<br>ACC.7 ← C<br>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 | $ACC \leftarrow ACC - [m] - \overline{C}$ |
| 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 | $[m] \leftarrow ACC - [m] - \overline{C}$ |
| 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<br>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<br>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 | $[m] \leftarrow [m] + 1$<br>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 | $ACC \leftarrow [m] + 1$<br>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 $[m].i \neq 0$ |
| 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 | $ACC \leftarrow ACC - [m]$ |
| 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 | $[m] \leftarrow ACC - [m]$ |
| 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 | $ACC \leftarrow ACC - x$ |
| 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 | $[m].3 \sim [m].0 \leftrightarrow [m].7 \sim [m].4$ |
| 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-EP (150mil) Outline Dimensions



THERMAL VARIATIONS ONLY
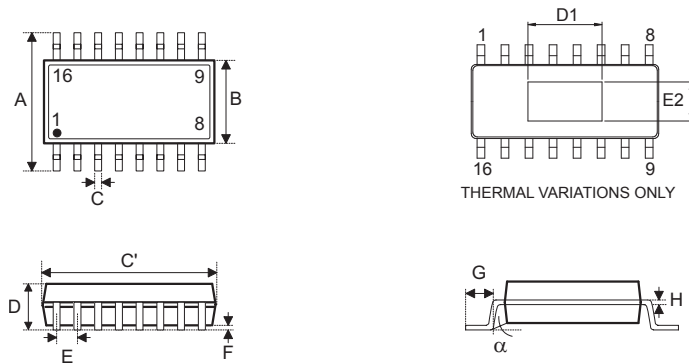
| Symbol | Dimensions in inch | | |
|---|---|---|---|
| | Min. | Nom. | Max. |
| A | — | 0.236 BSC | — |
| B | — | 0.154 BSC | — |
| C | 0.012 | — | 0.020 |
| C' | — | 0.390 BSC | — |
| D | — | — | 0.069 |
| D1 | 0.059 | — | — |
| E | — | 0.050 BSC | — |
| E2 | 0.039 | — | — |
| F | 0.000 | — | 0.006 |
| G | 0.016 | — | 0.050 |
| H | 0.004 | — | 0.010 |
| α | 0° | — | 8° |

| Symbol | Dimensions in mm | | |
|---|---|---|---|
| | Min. | Nom. | Max. |
| A | — | 6.00 BSC | — |
| B | — | 3.90 BSC | — |
| C | 0.31 | — | 0.51 |
| C' | — | 9.90 BSC | — |
| D | — | — | 1.75 |
| D1 | 1.50 | — | — |
| E | — | 1.27 BSC | — |
| E2 | 1.00 | — | — |
| F | 0.00 | — | 0.15 |
| G | 0.40 | — | 1.27 |
| H | 0.10 | — | 0.25 |
| α | 0° | — | 8° |

## 24-pin SSOP-EP (150mil) Outline Dimensions



THERMALLY ENHANCED VERIATIONS ONLY
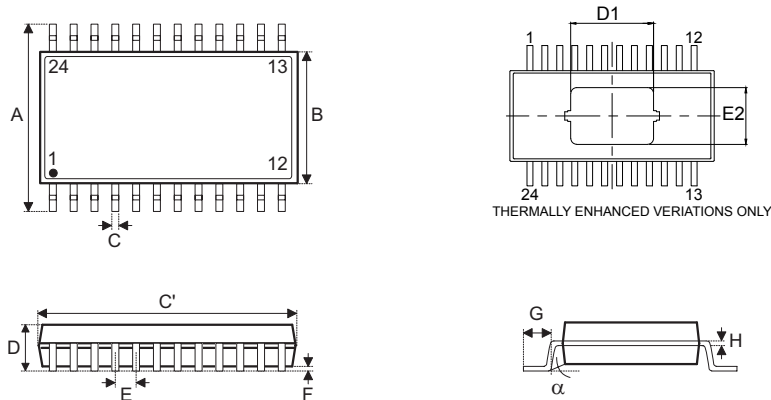
| Symbol | Dimensions in inch | | |
|--------|------|------|------|
| | Min. | Nom. | Max. |
| A | — | 0.236 BSC | — |
| B | — | 0.154 BSC | — |
| C | 0.008 | — | 0.012 |
| C' | — | 0.341 BSC | — |
| D | — | — | 0.069 |
| D1 | — | 0.140 | — |
| E | — | 0.025 BSC | — |
| E2 | — | 0.096 | — |
| F | 0.000 | — | 0.004 |
| G | 0.016 | — | 0.050 |
| H | 0.004 | — | 0.010 |
| α | 0° | — | 8° |

| Symbol | Dimensions in mm | | |
|--------|------|------|------|
| | Min. | Nom. | Max. |
| A | — | 6.00 BSC | — |
| B | — | 3.90 BSC | — |
| C | 0.20 | — | 0.30 |
| C' | — | 8.66 BSC | — |
| D | — | — | 1.75 |
| D1 | — | 3.56 | — |
| E | — | 0.635 BSC | — |
| E2 | — | 2.44 | — |
| F | 0.00 | — | 0.10 |
| G | 0.41 | — | 1.27 |
| H | 0.10 | — | 0.25 |
| α | 0° | — | 8° |

### SAW Type 24-pin QFN (4mm×4mm×0.75mm) Outline Dimensions
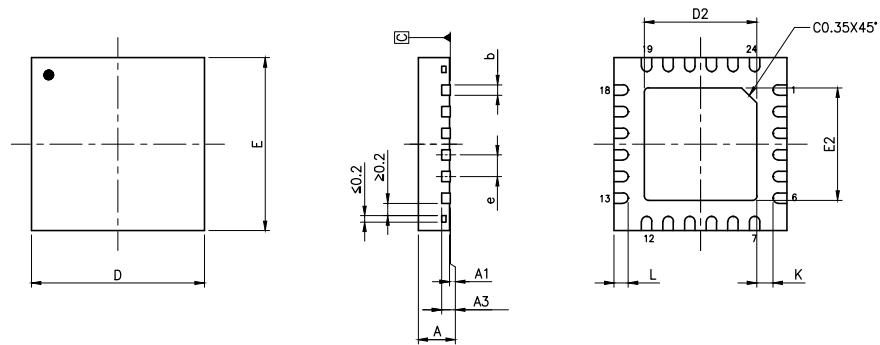


| Symbol | Dimensions in inch | | |
|---|---|---|---|
| | Min. | Nom. | Max. |
| A | 0.028 | 0.030 | 0.031 |
| A1 | 0.000 | 0.001 | 0.002 |
| A3 | — | 0.008 BSC | — |
| b | 0.007 | 0.010 | 0.012 |
| D | — | 0.157 BSC | — |
| E | — | 0.157 BSC | — |
| e | — | 0.020 BSC | — |
| D2 | 0.104 | 0.106 | 0.108 |
| E2 | 0.104 | 0.106 | 0.108 |
| L | 0.014 | 0.016 | 0.018 |

| Symbol | Dimensions in mm | | |
|---|---|---|---|
| | Min. | Nom. | Max. |
| A | 0.70 | 0.75 | 0.80 |
| A1 | 0.00 | 0.02 | 0.05 |
| A3 | — | 0.203 BSC | — |
| b | 0.18 | 0.25 | 0.30 |
| D | — | 4.00 BSC | — |
| E | — | 4.00 BSC | — |
| e | — | 0.50 BSC | — |
| D2 | 2.65 | 2.70 | 2.75 |
| E2 | 2.65 | 2.70 | 2.75 |
| L | 0.35 | 0.40 | 0.45 |