HT86BXX/HT86BRXX
Enhanced Voice 8-Bit MCU

Technical Document

- Application Note
  - HA0075E MCU Reset and Oscillator Circuits Application Note

Features

- Operating voltage: 2.2V~5.5V
- System clock: 4MHz~8MHz
- Crystal and RC system oscillator
- 16/20/24 I/O pins
- 8K×16-bit Program Memory
- 192×8/384×8-bit Data Memory
- External interrupt input
- Three 8-bit programmable Timers with overflow interrupt and 8-stage prescaler
- 12-bit high quality voltage type D/A output
- PWM circuit direct audio output
- External RC oscillator converter
- 8 capacitor/resistor sensor input
- Watchdog timer function
- 8-level subroutine nesting
- Low voltage reset function
- Integrated voice ROM with various capacities
- Power-down function and wake-up feature reduce power consumption
- Up to 0.5μs instruction cycle with 8MHz system clock at V_{DD}= 5V
- 63 powerful instructions

General Description

The Voice type series of MCUs are 8-bit high performance microcontrollers which include a voice synthesizer and tone generator. They are designed for applications which require multiple I/Os and sound effects, such as voice and melody. The devices can provide various sampling rates and beats, tone levels, tempos for speech synthesizer and melody generator. They also include an integrated high quality, voltage type DAC output. The external interrupt can be triggered with falling edges or both falling and rising edges.

The devices are excellent solutions for versatile voice and sound effect product applications with their efficient MCU instructions providing the user with programming capability for powerful custom applications. The system frequency can be up to 8MHz at an operating voltage of 2.2V and include a power-down function to reduce power consumption.

Device Types

Devices which have the letter ″BR″ within their part number, indicate that they are OTP devices offering the advantages of easy and effective program updates, using the Holtek range of development and programming tools. These devices provide the designer with the means for fast and low-cost product development cycles. Devices which have the letter ″B″ within their part number indicate that they are mask version devices. These devices offer a complementary device for applications that are at a mature state in their design process and have high volume and low cost demands.

Part numbers including ″R″ are OTP devices, all others are mask version devices.

Fully pin and functionally compatible with their OTP sister devices, the mask version devices provide the ideal substitute for products which have gone beyond their development cycle and are facing cost-down demands.

In this datasheet, for convenience, when describing device functions, only the OTP types are mentioned by name, however the same described functions also apply to the Mask type devices.

Selection Table

The devices include a comprehensive range of features, with most features common to all devices. The main features distinguishing them are Program Memory and Data Memory capacity, Voice ROM and Voice capacity, I/O count, stack size and package types. The functional differences between the devices are shown in the following table.

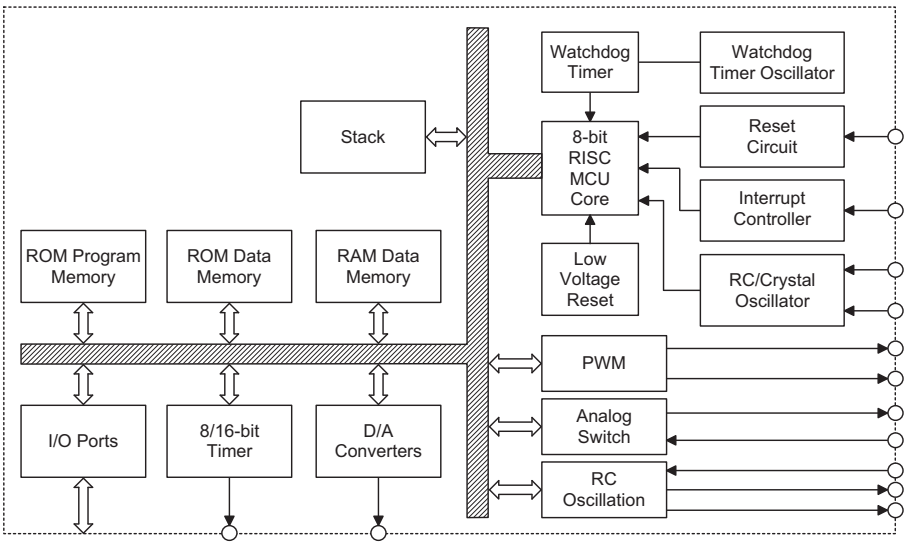
Part No.	VDD	Program Memory	Data Memory	Voice ROM	Voice Capacity	I/O	Timer		C/R-F	D/A	Stack	Package Types
							8-bit	16-bit				
HT86B03	2.2V~5.5V	4K×16	192×8	96K×8	36sec	12	3	—	—	12-bit	8	16NSOP, 24SSOP (150/209mil)
HT86BR10	2.2V~5.5V	8K×16	192×8	192K×8	72sec	16	3	—	—	12-bit	8	24SSOP(209mil), 28SOP, 44QFP
HT86B10												24SSOP (150/209mil), 28SOP, 44QFP
HT86B20	2.2V~5.5V	8K×16	192×8	256K×8	96sec	16	3	—	—	12-bit	8	28SOP, 44QFP
HT86BR30	2.2V~5.5V	8K×16	192×8	384K×8	144sec	16	3	—	—	12-bit	8	28SOP, 44QFP
HT86B30												
HT86B40	2.2V~5.5V	8K×16	384×8	512K×8	192sec	20	3	1	√	12-bit	8	28SOP, 44QFP
HT86B50	2.2V~5.5V	8K×16	384×8	768K×8	288sec	20	3	1	√	12-bit	8	28SOP, 44QFP
HT86BR60	2.2V~5.5V	8K×16	384×8	1024K×8	384sec	20	3	1	√	12-bit	8	28SOP
HT86B60												28SOP, 44QFP
HT86B70	2.2V~5.5V	8K×16	384×8	1536K×8	576sec	24	3	1	√	12-bit	8	44/100QFP
HT86B80	2.2V~5.5V	8K×16	384×8	2048K×8	768sec	24	3	1	√	12-bit	8	44/100QFP
HT86B90	2.2V~5.5V	8K×16	384×8	3072K×8	1152sec	24	3	1	√	12-bit	8	100QFP

Note: 1. For devices that exist in more than one package formats, the table reflects the situation for the larger package.

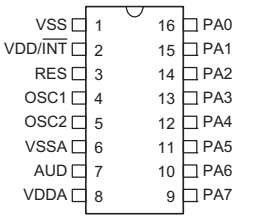
2. For the HT86B90, the operating voltage is 2.2V~5.5V at f_{SYS}=4MHz/3.3V~5.5V at f_{SYS}=8MHz.

3. Voice length is estimated by 21K-bit data rate

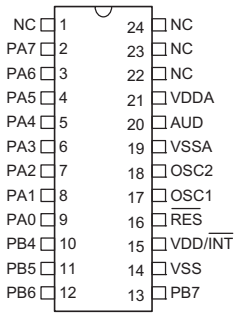
Block Diagram



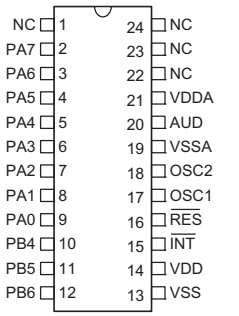
Pin Assignment



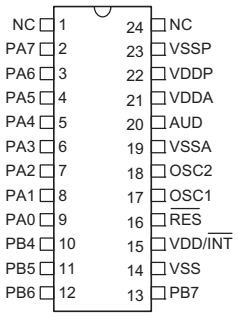
HT86B03 16 NSOP-A



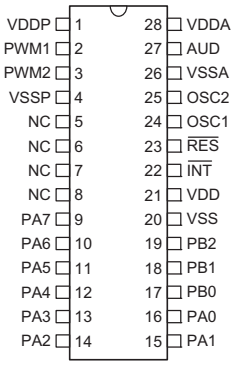
HT86B03 24 SSOP-A



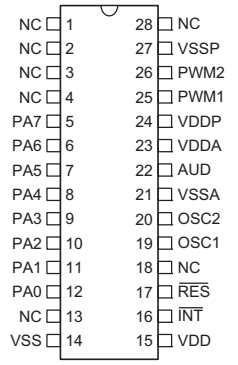
HT86B03 24 SSOP-B



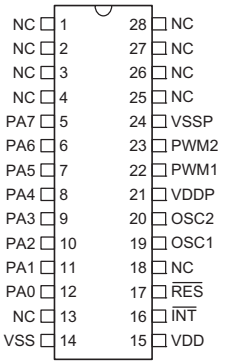
HT86B10/HT86BR10
24 SSOP-A



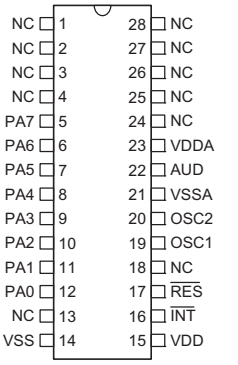
HT86B10 28 SOP-D
HT86B20/HT86B30 28 SOP-B
HT86BR10 28 SOP-B
HT86BR30 28 SOP-A



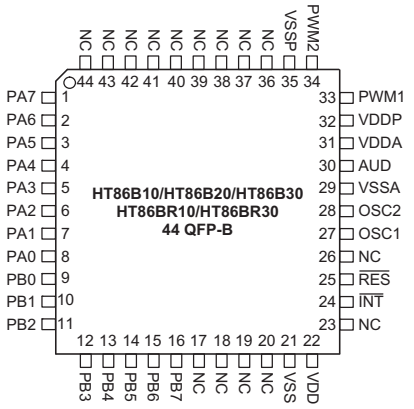
HT86B40/HT86B50/HT86B60
28 SOP-A



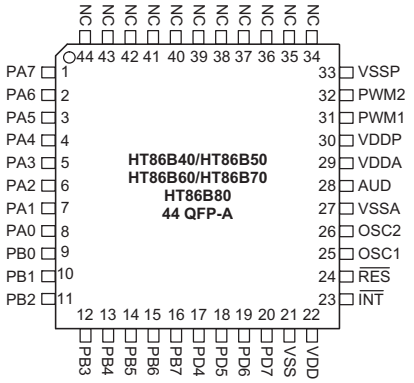
HT86B60 28 SOP-B
HT86BR60 28 SOP-A



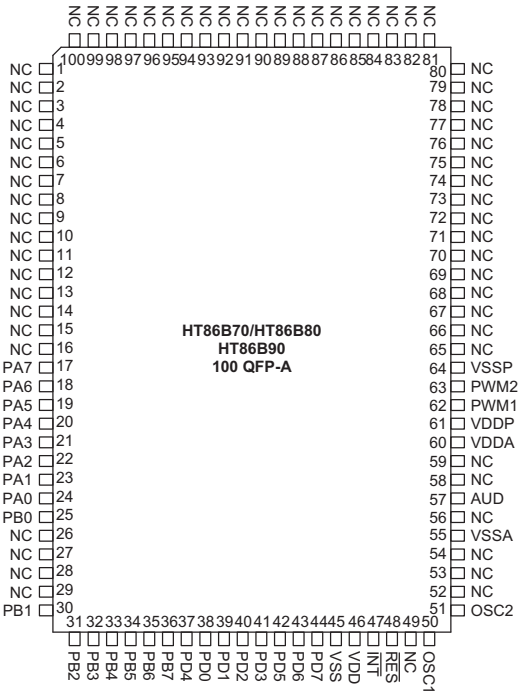
HT86B60 28 SOP-C
HT86BR60 28 SOP-B



HT86B10/HT86B20/HT86B30
HT86BR10/HT86BR30
44 QFP-B



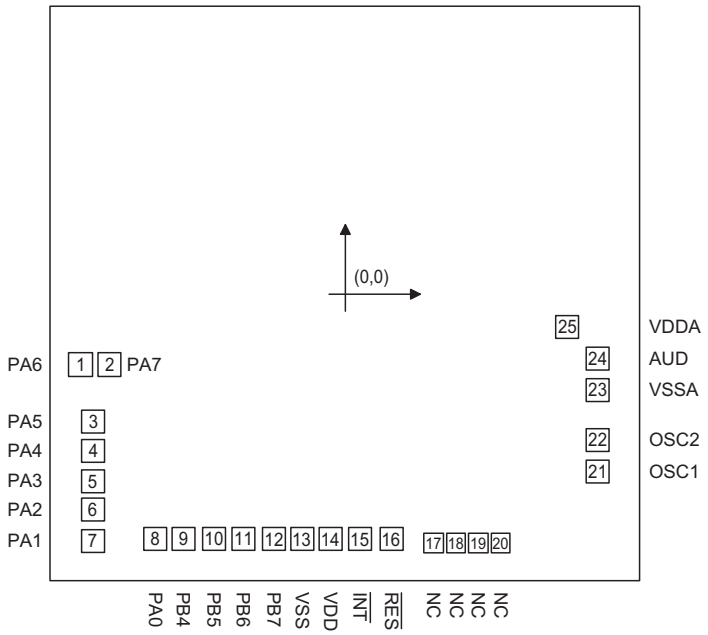
HT86B40/HT86B50
HT86B60/HT86B70
HT86B80
44 QFP-A



HT86B70/HT86B80
HT86B90
100 QFP-A

Pad Assignment

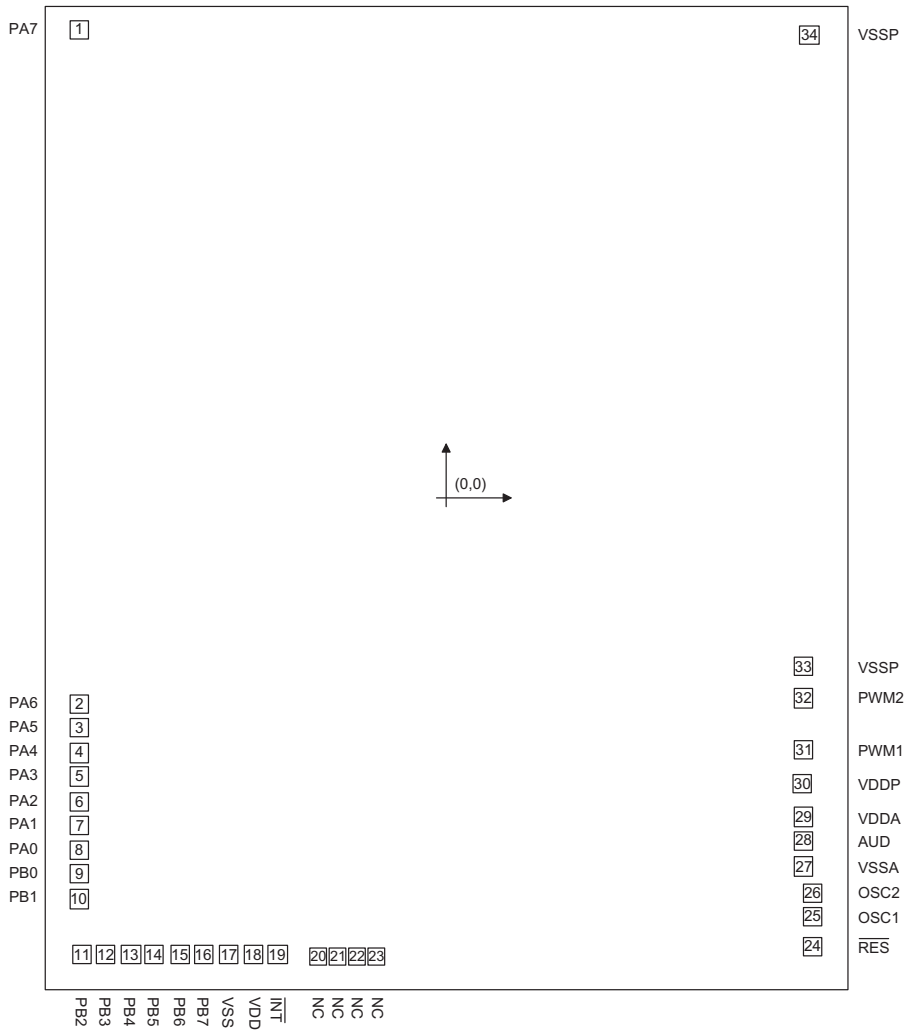
HT86B03



Chip size: 1975×1930 (μm)^2

* The IC substrate should be connected to VSS in the PCB layout artwork.

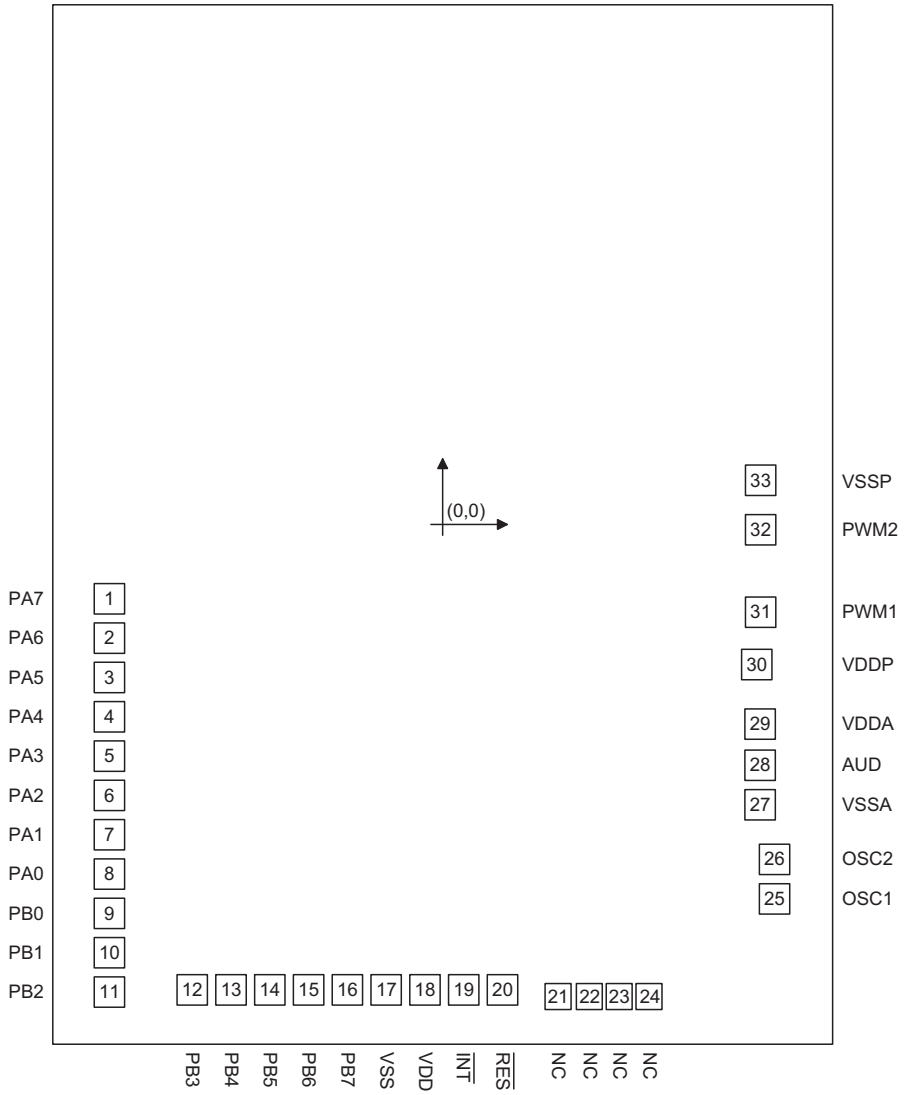
HT86BR10



Chip size: 3265×4010 (μm)^2

* The IC substrate should be connected to VSS in the PCB layout artwork.

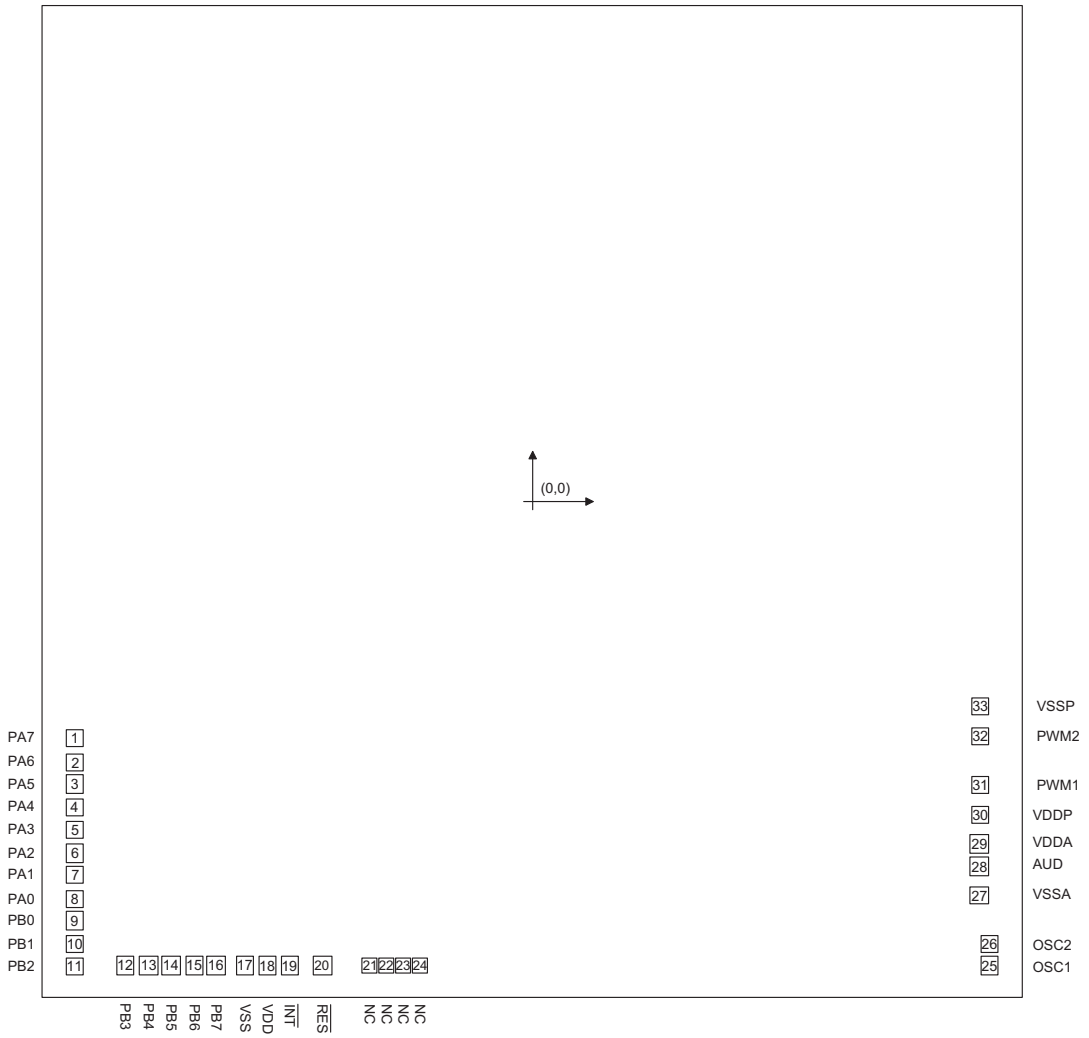
HT86B10



Chip size: 1975×2640 (μm)^2

* The IC substrate should be connected to VSS in the PCB layout artwork.

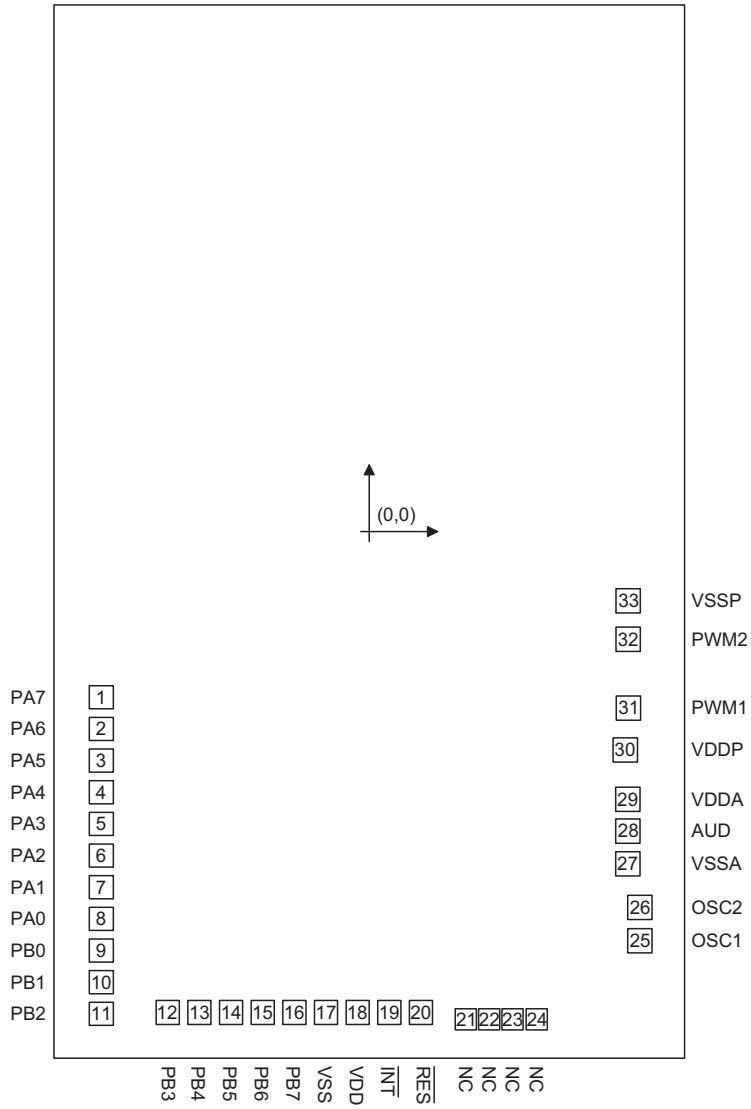
HT86BR30



Chip size: 4280×4330 (μm)^2

* The IC substrate should be connected to VSS in the PCB layout artwork.

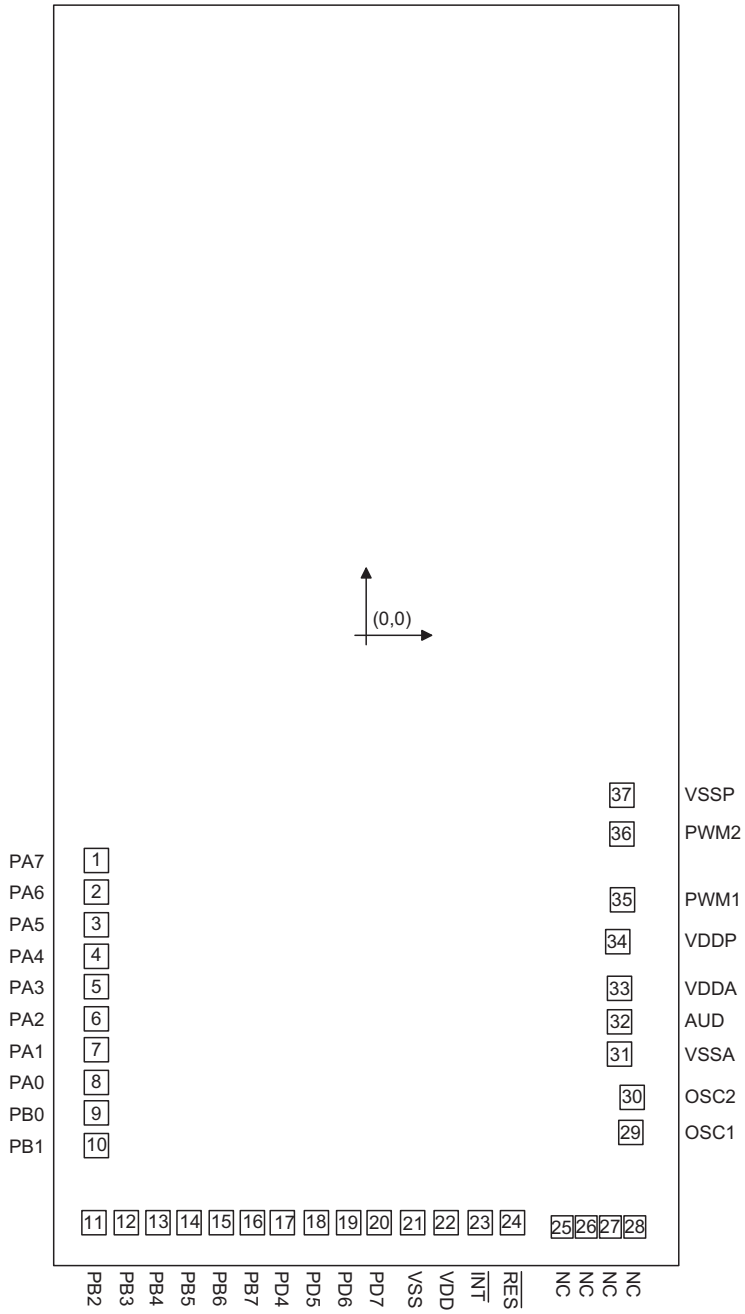
HT86B20/HT86B30



Chip size: 1975×3300 (μm)^2

* The IC substrate should be connected to VSS in the PCB layout artwork.

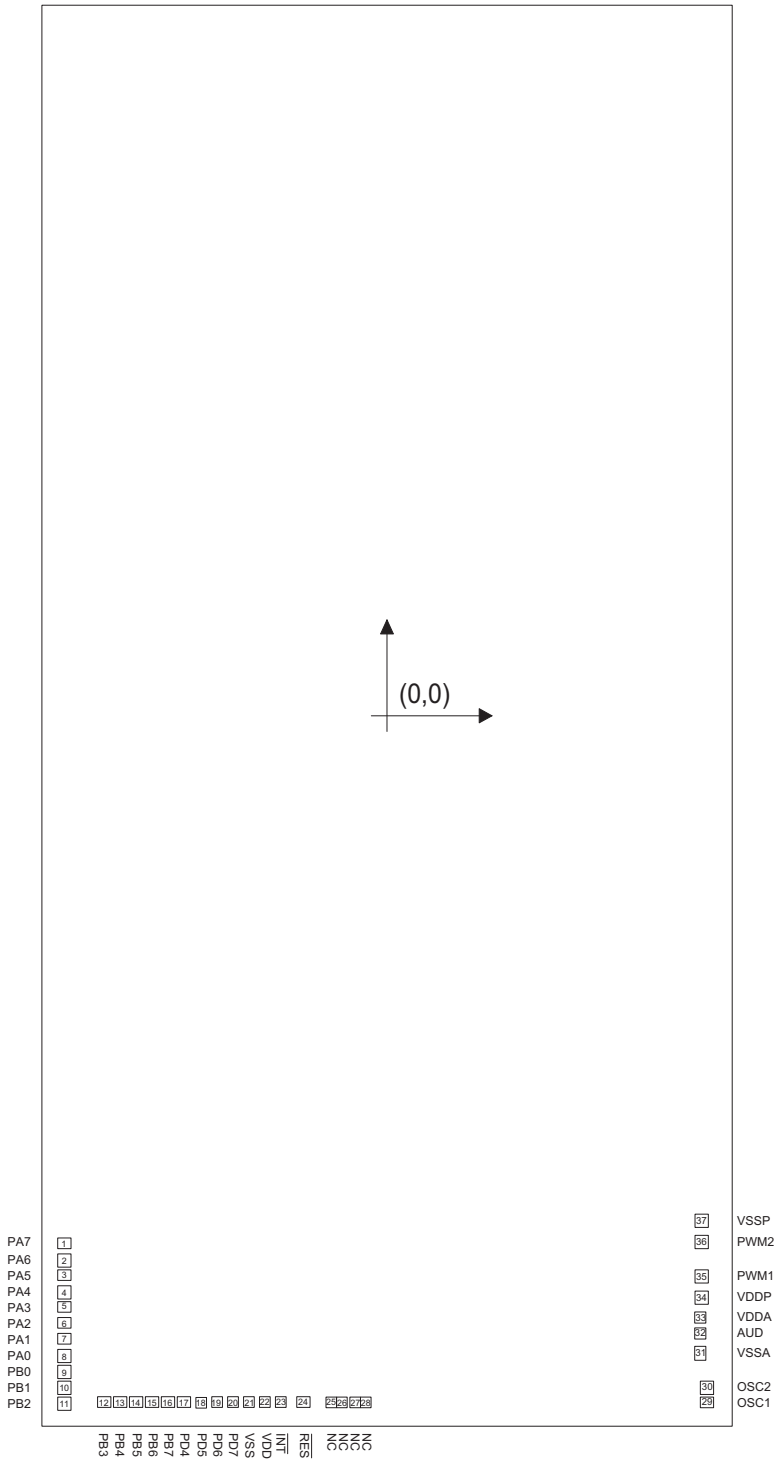
HT86B40



Chip size: 1975×3970 (μm)^2

* The IC substrate should be connected to VSS in the PCB layout artwork.

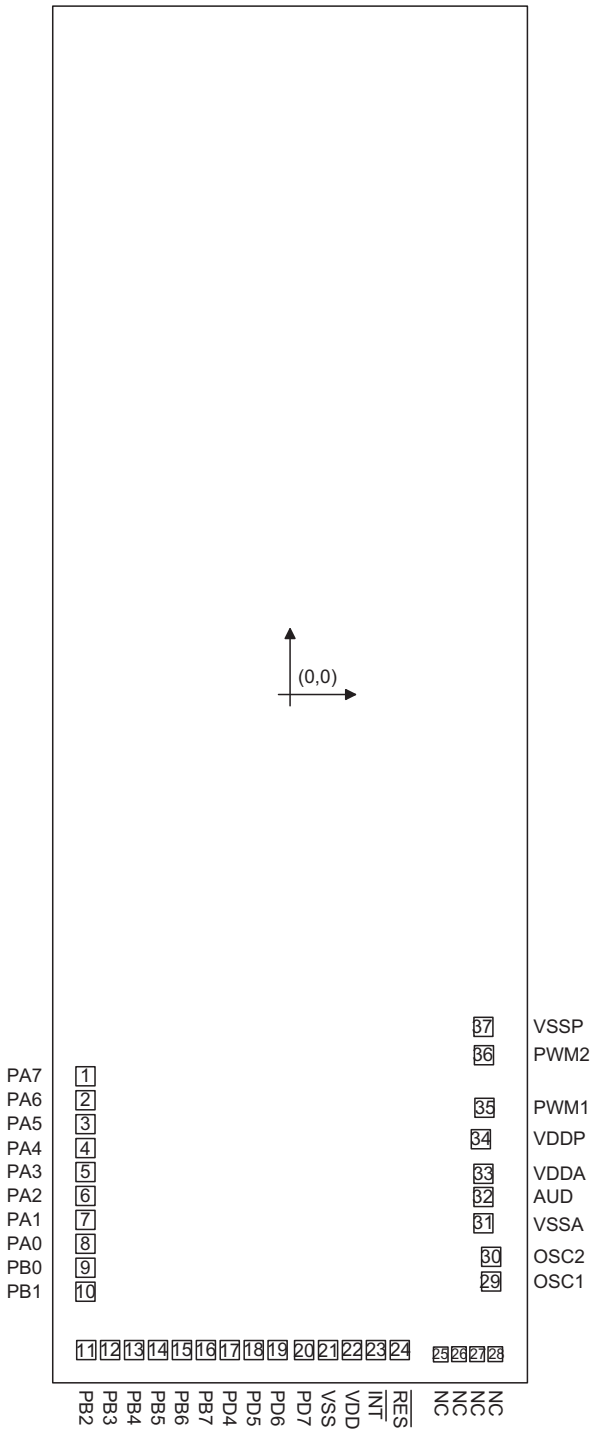
HT86BR60



Chip size: 4290×8835 (μm)^2

* The IC substrate should be connected to VSS in the PCB layout artwork.

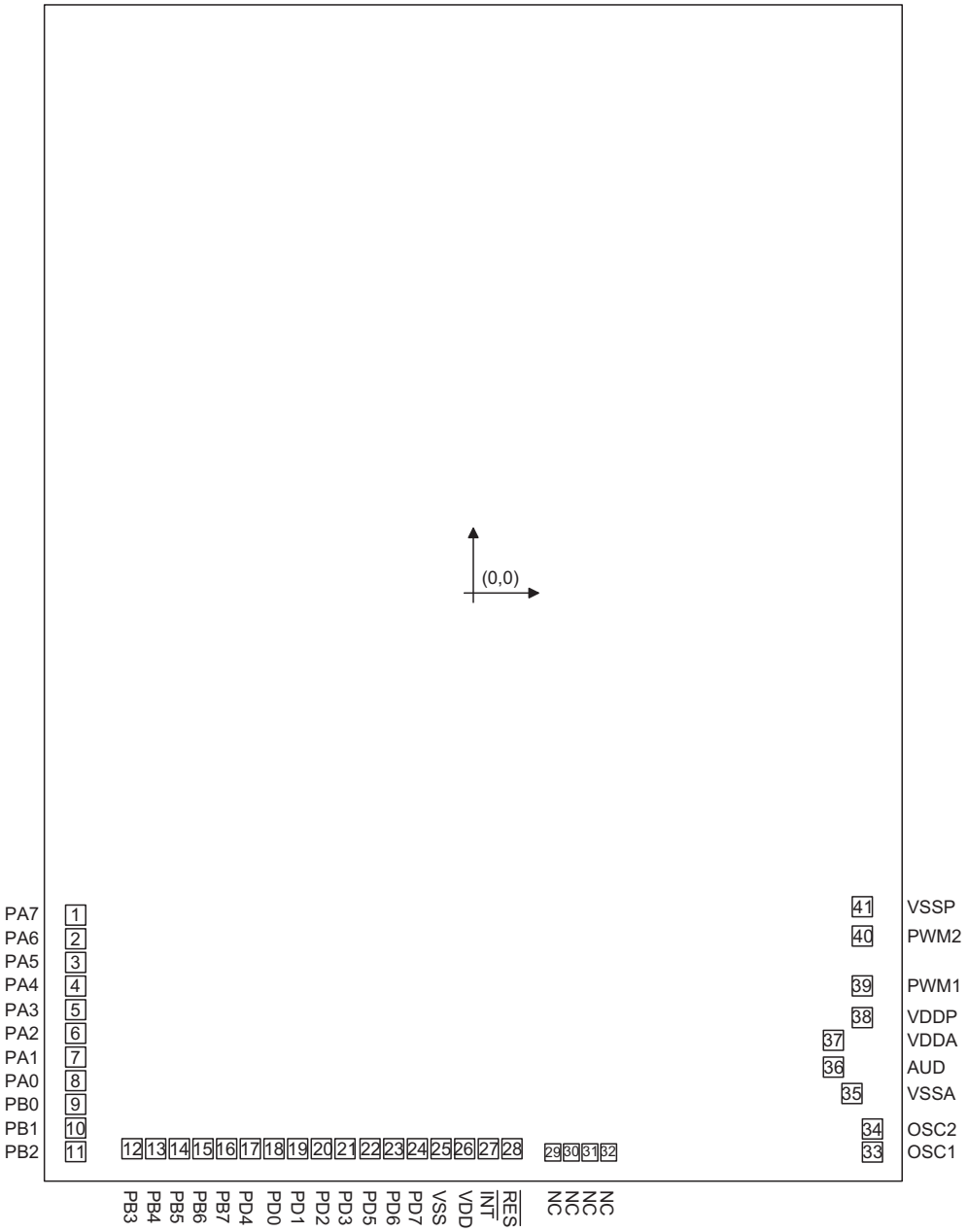
HT86B50/HT86B60



Chip size: 1975×5725 (μm)^2

* The IC substrate should be connected to VSS in the PCB layout artwork.

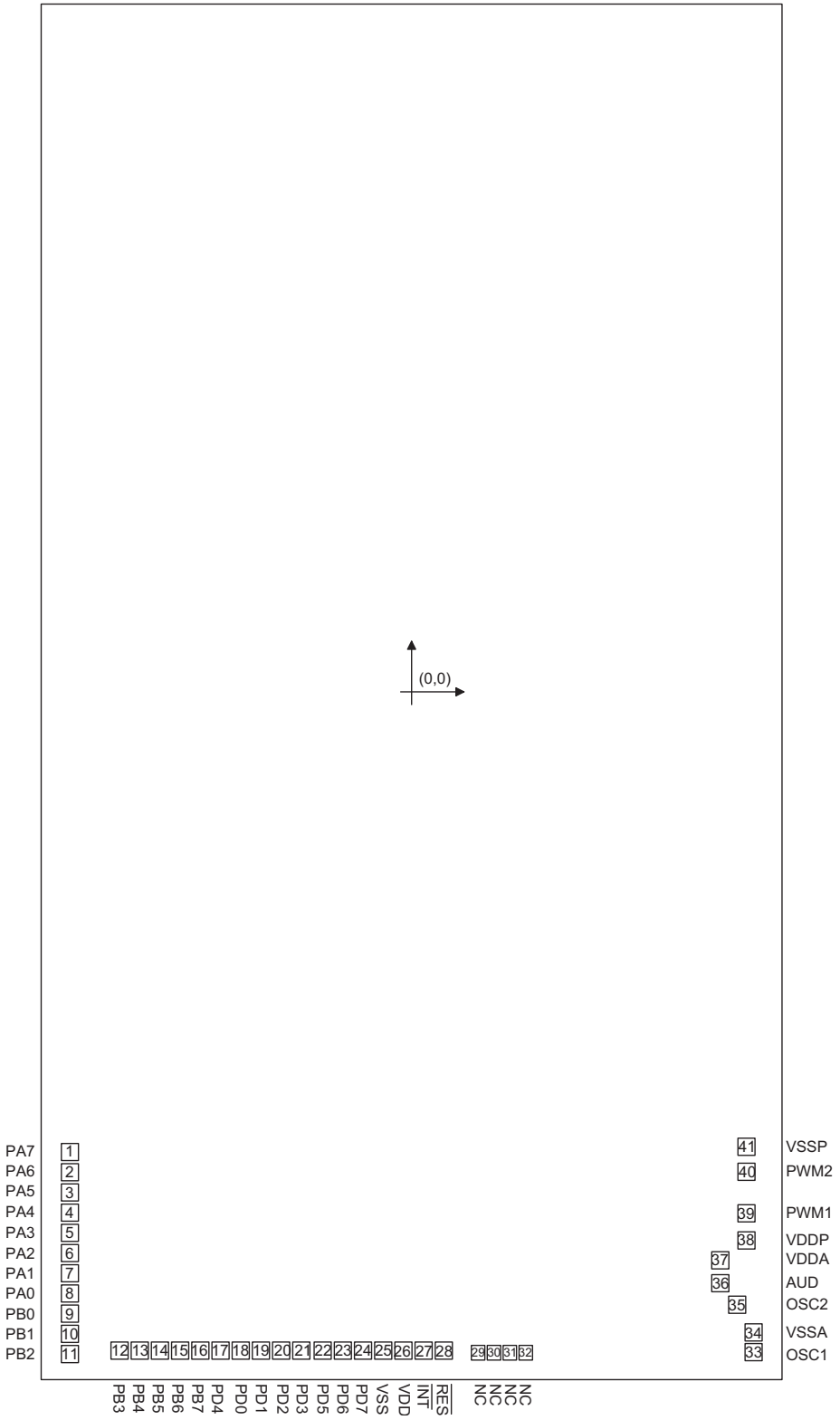
HT86B70/HT86B80



Chip size: 3615×4940 (μm)^2

* The IC substrate should be connected to VSS in the PCB layout artwork.

HT86B90



Chip size: 3620×6700 (μm)^2

* The IC substrate should be connected to VSS in the PCB layout artwork.

Pad Coordinates

HT86B03

Unit: μm

Pad No.	X	Y	Pad No.	X	Y
1	−879.400	−236.700	14	−46.350	−816.900
2	−784.400	−236.700	15	51.150	−816.900
3	−839.400	−428.200	16	154.150	−816.900
4	−839.400	−523.200	17	294.450	−833.650
5	−839.400	−626.200	18	368.450	−833.650
6	−839.400	−721.200	19	442.450	−833.650
7	−839.400	−824.200	20	516.450	−833.650
8	−632.350	−816.900	21	839.390	−592.550
9	−537.350	−816.900	22	839.390	−488.250
10	−434.350	−816.900	23	839.390	−321.808
11	−339.350	−816.900	24	839.390	−218.358
12	−236.350	−816.900	25	737.790	−116.308
13	−141.350	−816.900			

HT86BR10

Unit: μm

Pad No.	X	Y	Pad No.	X	Y
1	−1483.900	1900.000	18	−779.500	−1856.400
2	−1483.900	−838.050	19	−682.500	−1856.400
3	−1483.900	−933.050	20	−521.245	−1860.845
4	−1483.900	−1036.050	21	−447.245	−1860.845
5	−1483.900	−1131.050	22	−373.245	−1860.845
6	−1483.900	−1234.050	23	−299.245	−1860.845
7	−1483.900	−1329.050	24	1478.900	−1821.650
8	−1483.900	−1432.050	25	1478.900	−1700.550
9	−1483.900	−1527.050	26	1478.900	−1605.550
10	−1483.900	−1630.050	27	1442.800	−1497.530
11	−1474.850	−1856.400	28	1442.800	−1395.130
12	−1379.850	−1856.400	29	1442.800	−1295.470
13	−1276.850	−1856.400	30	1439.405	−1162.343
14	−1181.850	−1856.400	31	1442.395	−1024.550
15	−1078.850	−1856.400	32	1442.395	−814.050
16	−983.850	−1856.400	33	1442.395	−683.200
17	−881.645	−1856.400	34	1468.400	1879.850

HT86B10

Unit: μm

Pad No.	X	Y	Pad No.	X	Y
1	−839.400	−189.100	18	−45.150	−1171.900
2	−839.400	−284.100	19	50.850	−1171.900
3	−839.400	−387.100	20	153.850	−1171.900
4	−839.400	−482.100	21	294.450	−1188.650
5	−839.400	−585.100	22	368.450	−1188.650
6	−839.400	−680.100	23	442.450	−1188.650
7	−839.400	−783.100	24	516.450	−1188.650
8	−839.400	−878.100	25	838.940	−945.650
9	−839.400	−981.100	26	838.940	−843.250
10	−839.400	−1076.100	27	802.900	−704.400
11	−839.400	−1179.100	28	802.900	−601.500
12	−632.150	−1171.900	29	802.900	−504.300
13	−537.150	−1171.900	30	792.250	−351.400
14	−434.150	−1171.900	31	803.900	−218.050
15	−339.150	−1171.900	32	803.900	−7.550
16	−236.150	−1171.900	33	803.900	112.000
17	−141.150	−1171.900			

HT86BR30

Unit: μm

Pad No.	X	Y	Pad No.	X	Y
1	−1991.400	−1030.120	18	−1152.895	−2016.400
2	−1991.400	−1133.120	19	−1055.695	−2016.400
3	−1991.400	−1228.120	20	−913.745	−2016.400
4	−1991.400	−1331.120	21	−709.506	−2015.810
5	−1991.400	−1426.120	22	−635.506	−2015.810
6	−1991.400	−1529.120	23	−561.506	−2015.810
7	−1991.400	−1624.120	24	−487.506	−2015.810
8	−1991.400	−1727.120	25	1984.750	−2016.500
9	−1991.400	−1822.120	26	1984.750	−1921.500
10	−1991.400	−1925.120	27	1941.835	−1711.230
11	−1991.400	−2020.120	28	1941.835	−1586.960
12	−1771.750	−2016.400	29	1941.835	−1487.300
13	−1668.750	−2016.400	30	1946.850	−1363.920
14	−1573.750	−2016.400	31	1946.850	−1233.070
15	−1470.750	−2016.400	32	1946.850	−1022.570
16	−1375.750	−2016.400	33	1946.850	−891.720
17	−1251.695	−2016.780			

HT86B20/HT86B30

Unit: μm

Pad No.	X	Y	Pad No.	X	Y
1	−839.400	−519.100	18	−45.150	−1501.900
2	−839.400	−614.100	19	50.850	−1501.900
3	−839.400	−717.100	20	153.850	−1501.900
4	−839.400	−812.100	21	294.450	−1518.650
5	−839.400	−915.100	22	368.450	−1518.650
6	−839.400	−1010.100	23	442.450	−1518.650
7	−839.400	−1113.100	24	516.450	−1518.650
8	−839.400	−1208.100	25	838.940	−1275.650
9	−839.400	−1311.100	26	838.940	−1173.250
10	−839.400	−1406.100	27	802.900	−1034.400
11	−839.400	−1509.100	28	802.900	−931.500
12	−632.150	−1501.900	29	802.900	−834.300
13	−537.150	−1501.900	30	792.250	−681.400
14	−434.150	−1501.900	31	803.900	−548.050
15	−339.150	−1501.900	32	803.900	−337.550
16	−236.150	−1501.900	33	803.900	−218.000
17	−141.150	−1501.900			

HT86B40

Unit: μm

Pad No.	X	Y	Pad No.	X	Y
1	−839.400	−701.930	20	44.500	−1836.900
2	−839.400	−804.930	21	149.650	−1836.900
3	−839.400	−899.930	22	255.250	−1836.900
4	−839.400	−1002.930	23	359.150	−1836.900
5	−839.400	−1097.930	24	462.150	−1836.900
6	−839.400	−1200.930	25	619.850	−1853.600
7	−839.400	−1295.930	26	693.850	−1853.600
8	−839.400	−1398.930	27	767.850	−1853.600
9	−839.400	−1493.930	28	841.850	−1853.600
10	−839.400	−1596.930	29	839.390	−1551.700
11	−848.700	−1836.900	30	839.390	−1449.300
12	−745.700	−1836.900	31	802.900	−1311.300
13	−650.700	−1836.900	32	802.900	−1207.800
14	−547.700	−1836.900	33	802.900	−1103.500
15	−452.700	−1836.900	34	792.350	−959.450
16	−349.700	−1836.900	35	803.900	−829.350
17	−254.700	−1836.900	36	803.900	−618.850
18	−153.500	−1836.900	37	803.900	−499.300
19	−50.500	−1836.900			

HT86BR60

Unit: μm

Pad No.	X	Y	Pad No.	X	Y
1	−1996.400	−3279.080	20	954.350	−4269.280
2	−1996.400	−3382.080	21	853.150	−4269.280
3	−1996.400	−3477.080	22	753.150	−4268.900
4	−1996.400	−3580.080	23	657.150	−4268.900
5	−1996.400	−3675.080	24	515.200	−4268.900
6	−1996.400	−3778.080	25	345.100	−4268.850
7	−1996.400	−3873.080	26	271.100	−4268.850
8	−1996.400	−3976.080	27	197.100	−4268.850
9	−1996.400	−4074.580	28	123.100	−4268.850
10	−1996.400	−4177.580	29	1991.750	−4269.000
11	−1996.400	−4272.580	30	1991.750	−4174.000
12	−1750.825	−4269.280	31	1948.850	−3963.730
13	−1655.825	−4269.280	32	1948.850	−3839.460
14	−1552.825	−4269.280	33	1948.850	−3739.800
15	−1457.825	−4269.280	34	1953.850	−3616.420
16	−1354.825	−4269.280	35	1953.850	−3485.570
17	−1259.825	−4269.280	36	1953.850	−3275.070
18	−1152.350	−4269.280	37	1953.850	−3144.220
19	−1049.350	−4269.280			

HT86B50/HT86B60

Unit: μm

Pad No.	X	Y	Pad No.	X	Y
1	−839.400	−1579.430	20	44.600	−2714.400
2	−839.400	−1682.430	21	149.650	−2714.400
3	−839.400	−1777.430	22	255.250	−2714.400
4	−839.400	−1880.430	23	359.150	−2714.400
5	−839.400	−1975.430	24	462.150	−2714.400
6	−839.400	−2078.430	25	619.850	−2731.100
7	−839.400	−2173.430	26	693.850	−2731.100
8	−839.400	−2276.430	27	767.850	−2731.100
9	−839.400	−2371.430	28	841.850	−2731.100
10	−839.400	−2474.430	29	839.390	−2427.100
11	−848.700	−2714.400	30	839.390	−2326.800
12	−745.700	−2714.400	31	802.900	−2188.800
13	−650.700	−2714.400	32	802.900	−2085.300
14	−547.700	−2714.400	33	802.900	−1981.000
15	−452.700	−2714.400	34	792.350	−1836.950
16	−349.700	−2714.400	35	803.900	−1706.850
17	−254.700	−2714.400	36	803.900	−1496.350
18	−153.400	−2714.400	37	803.900	−1376.800
19	−50.400	−2714.400			

HT86B70/HT86B80

Unit: μm

Pad No.	X	Y	Pad No.	X	Y
1	−1659.400	−1342.900	22	−431.055	−2321.900
2	−1659.400	−1437.900	23	−328.055	−2321.900
3	−1659.400	−1540.900	24	−233.055	−2321.900
4	−1659.400	−1635.900	25	−130.855	−2321.900
5	−1659.400	−1738.900	26	−32.865	−2321.900
6	−1659.400	−1833.900	27	67.140	−2321.900
7	−1659.400	−1936.900	28	170.140	−2321.900
8	−1659.400	−2031.900	29	332.095	−2327.150
9	−1659.400	−2134.900	30	406.095	−2327.150
10	−1659.400	−2229.900	31	480.095	−2327.150
11	−1659.400	−2332.900	32	554.095	−2327.150
12	−1419.255	−2321.900	33	1658.950	−2324.995
13	−1324.255	−2321.900	34	1658.950	−2229.995
14	−1221.255	−2321.900	35	1576.095	−2087.795
15	−1126.255	−2321.900	36	1495.595	−1979.695
16	−1023.255	−2321.900	37	1495.595	−1869.845
17	−928.255	−2321.900	38	1623.910	−1774.245
18	−825.255	−2321.900	39	1623.910	−1640.895
19	−730.255	−2321.900	40	1623.910	−1430.395
20	−627.255	−2321.900	41	1623.910	−1310.845
21	−532.255	−2321.900			

HT86B90

Unit: μm

Pad No.	X	Y	Pad No.	X	Y
1	−1661.900	−2222.900	22	−433.555	−3201.900
2	−1661.900	−2317.900	23	−330.555	−3201.900
3	−1661.900	−2420.900	24	−235.555	−3201.900
4	−1661.900	−2515.900	25	−133.355	−3201.900
5	−1661.900	−2618.900	26	−35.365	−3201.900
6	−1661.900	−2713.900	27	64.640	−3201.900
7	−1661.900	−2816.900	28	167.640	−3201.900
8	−1661.900	−2911.900	29	329.595	−3207.150
9	−1661.900	−3014.900	30	403.595	−3207.150
10	−1661.900	−3109.900	31	477.595	−3207.150
11	−1661.900	−3212.900	32	551.595	−3207.150
12	−1421.755	−3201.900	33	1656.900	−3204.995
13	−1326.755	−3201.900	34	1493.095	−2859.695
14	−1223.755	−3201.900	35	1573.595	−2967.795
15	−1128.755	−3201.900	36	1656.900	−3109.995
16	−1025.755	−3201.900	37	1493.095	−2749.845
17	−930.755	−3201.900	38	1621.410	−2654.245
18	−827.755	−3201.900	39	1621.410	−2520.895
19	−732.755	−3201.900	40	1621.410	−2310.395
20	−629.755	−3201.900	41	1621.410	−2190.845
21	−534.755	−3201.900			

Pin Description

HT86B03/HT86B10/HT86B20/HT86B30/HT86BR10/HT86BR30

Pad Name	I/O	Options	Description
PA0~PA7	I/O	Wake-up, Pull-high or None	Bidirectional 8-bit I/O port. Software instructions determined the CMOS output or Schmitt trigger with a pull-high resistor (determined by option).
PB0~PB7	I/O	Pull-high or None	Bidirectional 8-bit I/O port. Software instructions determined the CMOS output or Schmitt trigger with a pull-high resistor (determined by option). The HT86B03 device only has PB4~PB7 port pins.
AUD	O	—	Audio output for driving an external transistor or for driving HT82V733
PWM1 PWM2	O	—	Audio PWM outputs. The HT86B03 has no PWM outputs.
$\overline{RES}$	I	—	Schmitt trigger reset input. Active low.
$\overline{INT}$	I	Falling Edge Trigger or Falling/Rising Edge Trigger	External interrupt Schmitt trigger input without pull-high resistor. A configuration option determines if the interrupt active edge is a falling edge only or both a falling and rising edge. Falling edge triggered active on a high to low transition. Rising edge triggered active on a low to high transition. Input voltage is the same as operating voltage.
OSC1 OSC2	—	Crystal or RC	OSC1, OSC2 are connected to an external RC network or external crystal, determined by configuration option, for the internal system clock. If the RC system clock option is selected, pin OSC2 can be used to measure the system clock at 1/4 frequency.
VDD	—	—	Positive digital power supply
VSS	—	—	Negative digital power supply, ground.
VDDA	—	—	Positive DAC circuit power supply
VSSA	—	—	Negative DAC circuit power supply, ground.
VDDP	—	—	Positive audio PWM circuit power supply
VSSP	—	—	Negative audio PWM circuit power supply, ground.

Note: 1. Each pin on PA can be programmed through a configuration option to have a wake-up function.

2. Individual pins can be selected to have pull-high resistors.

HT86B40/HT86B50/HT86B60/HT86BR60

Pad Name	I/O	Options	Description
PA0~PA7	I/O	Wake-up, Pull-high or None	Bi-directional 8-bit I/O port. Software instructions determined the CMOS output or Schmitt trigger with a pull-high resistor (determined by option).
PB0~PB7/ K0~K7	I/O	Pull-high or None	Bi-directional 8-bit I/O port. Software instructions determined the CMOS output or Schmitt trigger with a pull-high resistor (determined by option). Pins PB0~PB7 are pin-shared with C/R-F input pins K0~K7.
PD4/RCOUT PD5/RR PD6/RC PD7/CC	I/O	Pull-high or None	Bi-directional 4-bit I/O port. Software instructions determined the CMOS output or Schmitt trigger with a pull-high resistor (determined by option). Pins PD4~PD7 are pin-shared with R/F OSC input pins RR, RC and CC. RCOUT: Capacitor or resistor connection pin to RC OSC for input. RR: Oscillation input pin RC: Reference resistor connection pin for output CC: Reference capacitor connection pin for output
AUD	O	—	Audio output for driving an external transistor or for driving HT82V733
PWM1 PWM2	O	—	Audio PWM outputs
$\overline{RES}$	I	—	Schmitt trigger reset input. Active low.
$\overline{INT}$	I	Falling Edge Trigger or Falling/Rising Edge Trigger	External interrupt Schmitt trigger input without pull-high resistor. A configuration option determines if the interrupt active edge is a falling edge only or both a falling and rising edge. Falling edge triggered active on a high to low transition. Rising edge triggered active on a low to high transition. Input voltage is the same as operating voltage.
OSC1 OSC2	—	Crystal or RC	OSC1, OSC2 are connected to an external RC network or external crystal, determined by configuration option, for the internal system clock. If the RC system clock option is selected, pin OSC2 can be used to measure the system clock at 1/4 frequency.
VDD	—	—	Positive digital power supply
VSS	—	—	Negative digital power supply, ground.
VDDA	—	—	Positive DAC circuit power supply
VSSA	—	—	Negative DAC circuit power supply, ground.
VDDP	—	—	Positive audio PWM circuit power supply
VSSP	—	—	Negative audio PWM circuit power supply, ground.

Note: 1. Each pin on PA can be programmed through a configuration option to have a wake-up function.
2. Individual pins can be selected to have pull-high resistors.

HT86B70/HT86B80/HT86B90

Pad Name	I/O	Options	Description
PA0~PA7	I/O	Wake-up, Pull-high or None	Bi-directional 8-bit I/O port. Software instructions determined the CMOS output or Schmitt trigger with a pull-high resistor (determined by option).
PB0~PB7/ K0~K7	I/O	Pull-high or None	Bi-directional 8-bit I/O port. Software instructions determined the CMOS output or Schmitt trigger with a pull-high resistor (determined by option). Pins PB0~PB7 are pin-shared with C/R-F input pins K0~K7.
PD0~PD3 PD4/RCOUT PD5/RR PD6/RC PD7/CC	I/O	Pull-high or None	Bi-directional 8-bit I/O port. Software instructions determined the CMOS output or Schmitt trigger with a pull-high resistor (determined by option). Pins PD4~PD7 are pin-shared with R/F OSC input pins RR, RC and CC. RCOUT: Capacitor or resistor connection pin to RC OSC for input. RR: Oscillation input pin RC: Reference resistor connection pin for output CC: Reference capacitor connection pin for output
AUD	O	—	Audio output for driving an external transistor or for driving HT82V733
PWM1 PWM2	O	—	Audio PWM outputs
$\overline{RES}$	I	—	Schmitt trigger reset input. Active low.
$\overline{INT}$	I	Falling Edge Trigger or Falling/Rising Edge Trigger	External interrupt Schmitt trigger input without pull-high resistor. A configuration option determines if the interrupt active edge is a falling edge only or both a falling and rising edge. Falling edge triggered active on a high to low transition. Rising edge triggered active on a low to high transition. Input voltage is the same as operating voltage.
OSC1 OSC2	—	Crystal or RC	OSC1, OSC2 are connected to an external RC network or external crystal, determined by configuration option, for the internal system clock. If the RC system clock option is selected, pin OSC2 can be used to measure the system clock at 1/4 frequency.
VDD	—	—	Positive digital power supply
VSS	—	—	Negative digital power supply, ground.
VDDA	—	—	Positive DAC circuit power supply
VSSA	—	—	Negative DAC circuit power supply, ground.
VDDP	—	—	Positive audio PWM circuit power supply
VSSP	—	—	Negative audio PWM circuit power supply, ground.

Note: 1. Each pin on PA can be programmed through a configuration option to have a wake-up function.
2. Individual pins can be selected to have pull-high resistors.

Absolute Maximum Ratings

Supply Voltage ..... V_{SS}+2.2V to V_{SS}+5.5V
Input Voltage ..... V_{SS}–0.3V to V_{DD}+0.3V
I_{OL} Total 150mA
Total Power Dissipation 500mW

Storage Temperature ..... –50°C to 125°C
Operating Temperature ..... –40°C to 85°C
I_{OH} Total ..... –100mA

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

D.C. Characteristics

Ta=25°C

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
V_{DD}	Operating Voltage	—	f_{SYS}=4MHz/8MHz	2.2	—	5.5	V
			f_{SYS}=4MHz for HT86B90 only	2.2	—	5.5	V
			f_{SYS}=8MHz for HT86B90 only	3.3	—	5.5	V
I_{DD}	Operating Current	3V	No load, f_{SYS}=4MHz, DAC/PWM disable	—	—	1.5	mA
		5V		—	—	5	mA
		3V	No load, f_{SYS}=8MHz, DAC/PWM disable	—	—	3	mA
		5V		—	—	7	mA
I_{STB1}	Standby Current (WDT Off)	3V	No load, system HALT WDT disable	—	—	1	μA
		5V		—	—	2	μA
I_{STB2}	Standby Current (WDT On)	3V	No load, system HALT WDT enable	—	—	7	μA
		5V		—	—	10	μA
V_{IL1}	Input Low Voltage for I/O Ports	—	—	0	—	$0.3V_{DD}$	V
V_{IH1}	Input High Voltage for I/O Ports	—	—	$0.7V_{DD}$	—	V_{DD}	V
V_{IL2}	Input Low Voltage ($\overline{RES}$)	—	—	0	—	$0.4V_{DD}$	V
V_{IH2}	Input High Voltage ($\overline{RES}$)	—	—	$0.9V_{DD}$	—	V_{DD}	V
V_{IL3}	Input Low Voltage for EXT INT	—	—	0	—	$0.3V_{DD}$	V
V_{IH3}	Input High Voltage for EXT INT	—	—	$0.7V_{DD}$	—	V_{DD}	V
V_{LVR}	Low Voltage Reset	—	LVR 2.2V option	2.1	2.2	2.3	V
I_{OL1}	I/O Port Sink Current	3V	V_{OL}=0.1V_{DD}	4	—	—	mA
		5V		10	—	—	mA
I_{OH1}	I/O Port Source Current	3V	V_{OH}=0.9V_{DD}	–2	—	—	mA
		5V		–5	—	—	mA
I_{OL2}	RC and CC Sink Current	3V	V_{OL}=0.1V_{DD}	4	—	—	mA
		5V		10	—	—	mA
I_{OH2}	RC and CC Source Current	3V	V_{OH}=0.9V_{DD}	–2	—	—	mA
		5V		–5	—	—	mA
I_{OL3}	PWM1/PWM2 Sink Current	3V	V_{OL}=0.1V_{DD}	50	—	—	mA
		5V		80	—	—	mA
I_{OH3}	PWM1/PWM2 Source Current	3V	V_{OH}=0.9V_{DD}	–14.5	—	—	mA
		5V		–26	—	—	mA
I_{AUD}	AUD Source Current	3V	V_{OH}=0.9V_{DD}	–1.5	—	—	mA
		5V		–3	—	—	mA
R_{PH}	Pull-high Resistance	3V	—	20	60	100	kΩ
		5V		10	30	50	kΩ

A.C. Characteristics

Ta=25°C

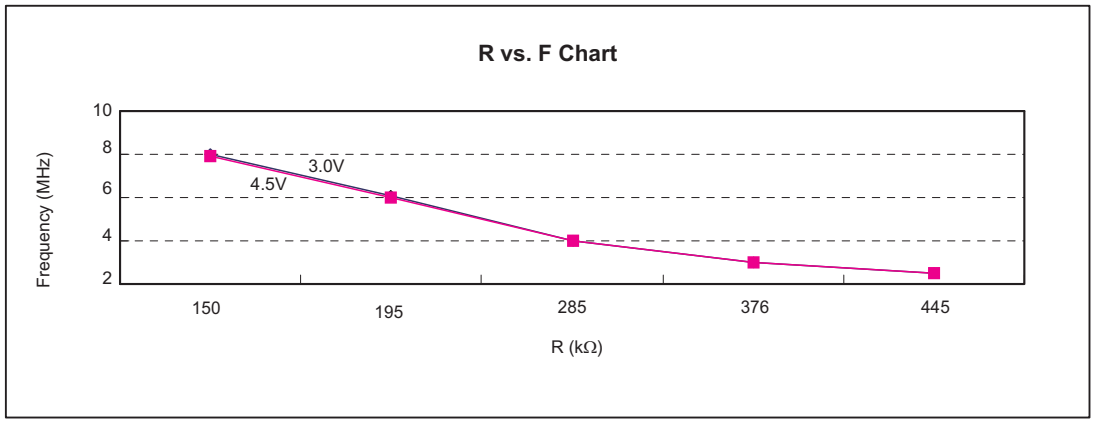
Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
f_{SYS}	System Clock (RC OSC, Crystal OSC)	—	2.2V~5.5V	4	—	8	MHz
t_{WDTOSC}	Watchdog Oscillator Period	3V	—	45	90	180	μs
		5V	—	32	65	130	μs
t_{RES}	External Reset Low Pulse Width	—	—	1	—	—	μs
t_{SST}	System Start-up Timer Period	—	Wake-up from HALT	—	1024	—	*t_{SYS}
t_{LVR}	Low Voltage Reset Time	—	—	2	—	—	ms
t_{INT}	Interrupt Pulse Width	—	—	1	—	—	μs
t_{MAT}	Circumscribe Memory Access Time	—	2.2V~5.5V	—	—	400	ns

Note: *t_{SYS}=1/f_{SYS}

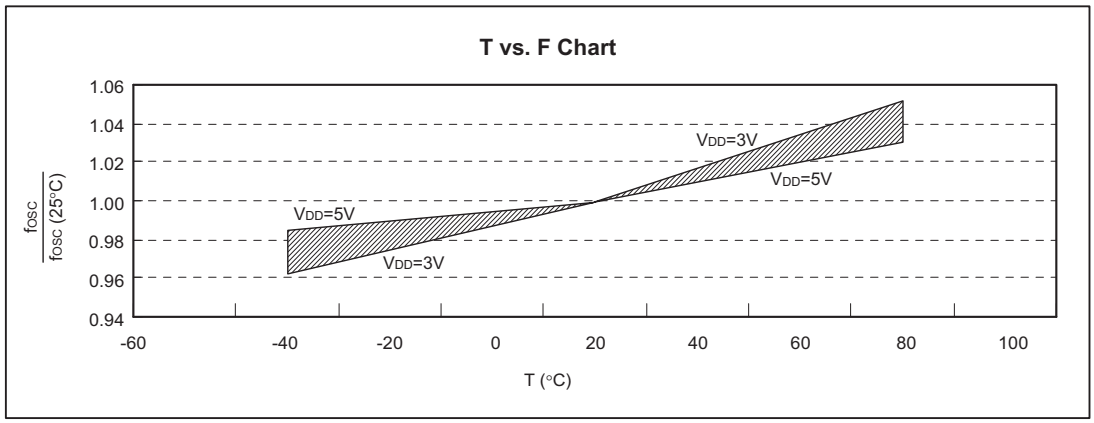
Characteristics Curves

HT86BRxx

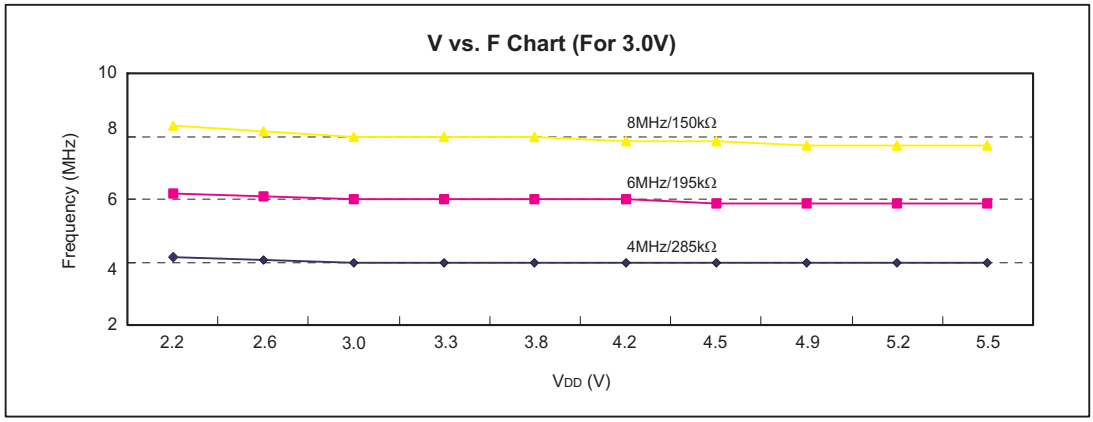
- R vs. F Chart Characteristics Curves



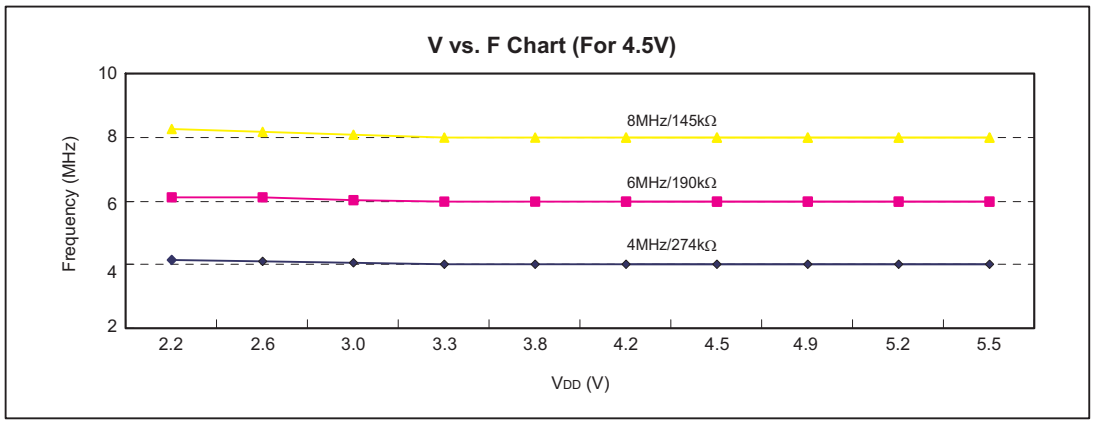
- T vs. F Chart Characteristics Curves



• V vs. F Chart Characteristics Curves – 3.0V

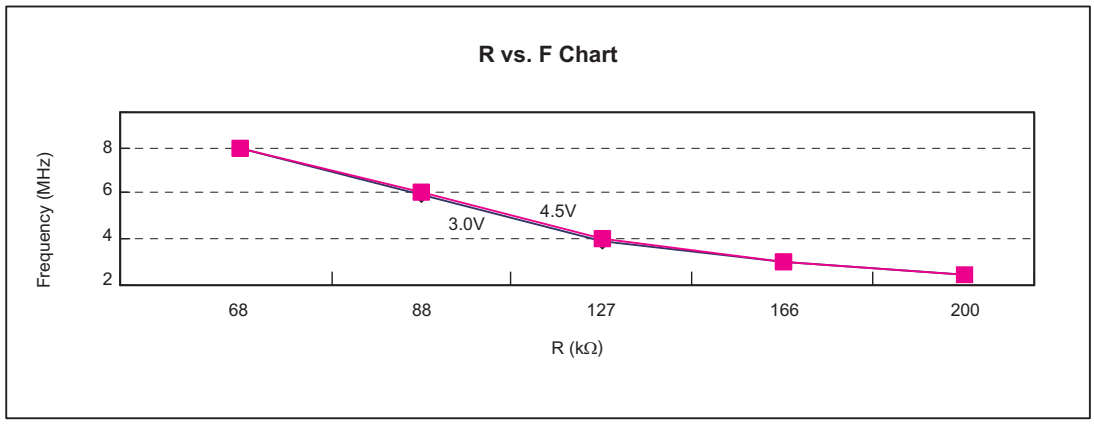


• V vs. F Chart Characteristics Curves – 4.5V

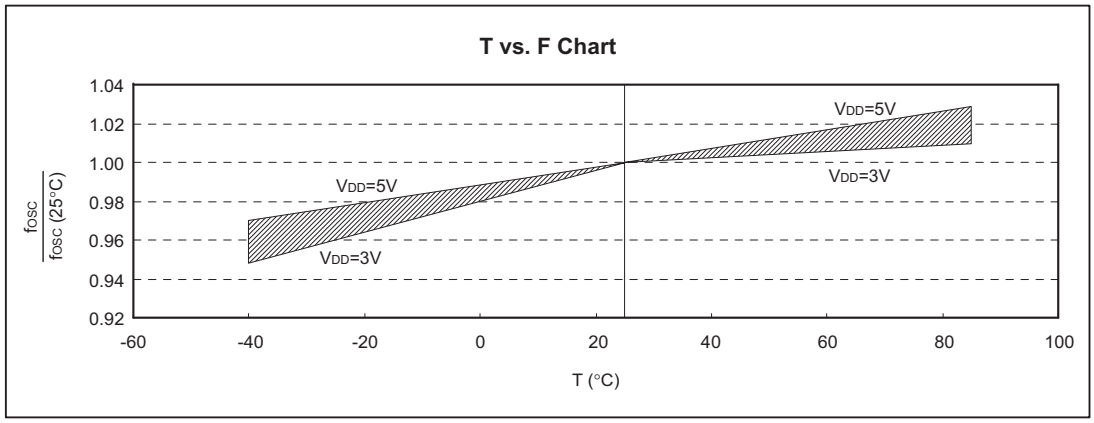


HT86Bxx

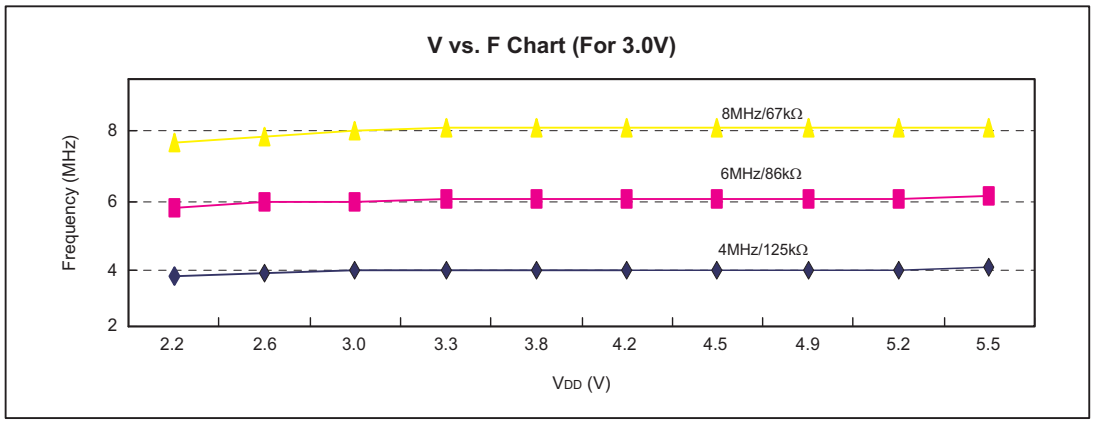
• R vs. F Chart Characteristics Curves



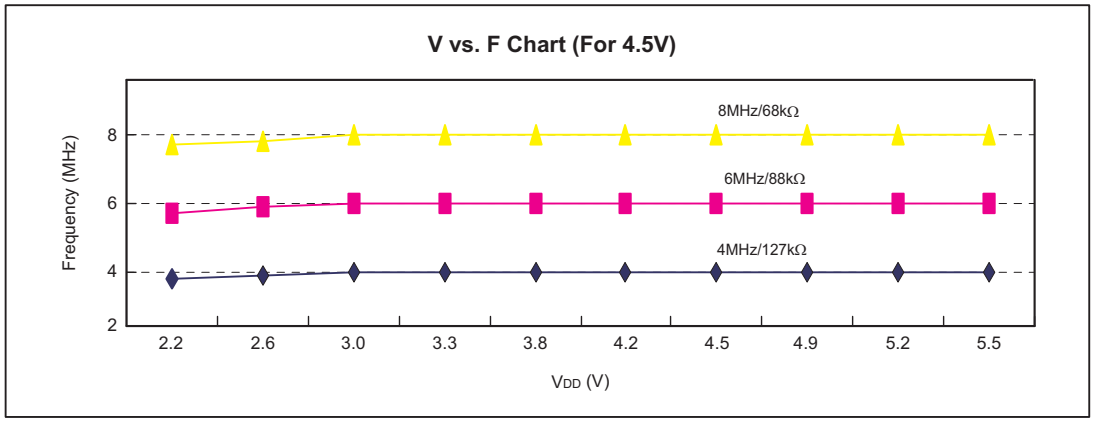
- T vs. F Chart Characteristics Curves



- V vs. F Chart Characteristics Curves – 3.0V



- V vs. F Chart Characteristics Curves – 4.5V



System Architecture

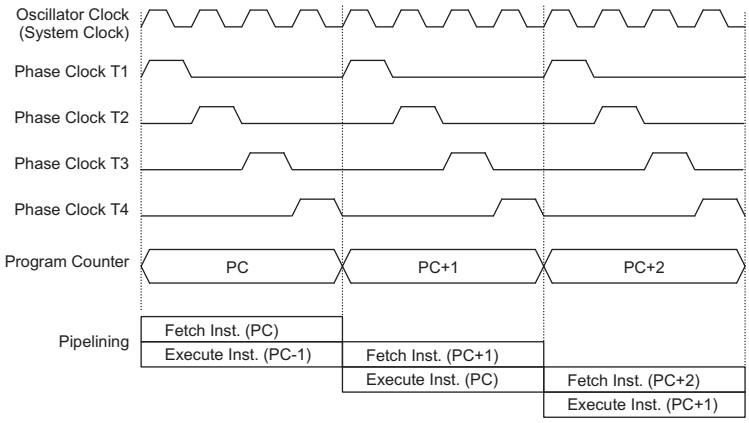
A key factor in the high-performance features of the Holtek range of Voice microcontrollers is attributed to the internal system architecture. The range of devices take advantage of the usual features found within RISC microcontrollers providing increased speed of operation and enhanced performance. 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. An 8-bit wide ALU is used in practically all operations of the instruction set. It 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, voltage type DAC, PWM direct drive output, capacitor/resistor sensor input and external RC oscillator converter with maximum reliability and flexibility.

Clocking and Pipelining

The main system clock, derived from either a Crystal/ Resonator or RC 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.

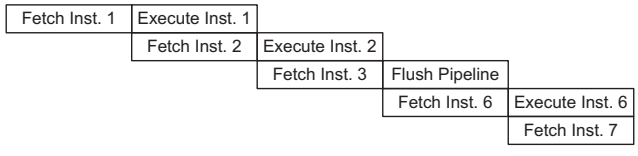
When the RC oscillator is used, OSC2 is freed for use as a T1 phase clock synchronizing pin. This T1 phase clock has a frequency of $f_{SYS}/4$ with a 1:3 high/low duty cycle.

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
```



Instruction Fetching

Program Counter

During program execution, the Program Counter is used to keep track of the address of the next instruction to be executed. It is automatically incremented by one each time an instruction is executed except for instructions, such as ″JMP″ or ″CALL″, that demand a jump to a non-consecutive Program Memory address. Note that the Program Counter width varies with the Program Memory capacity depending upon which device is selected. However, it must be noted that only the lower 8 bits, known as the Program Counter Low Register, are directly addressable by user.

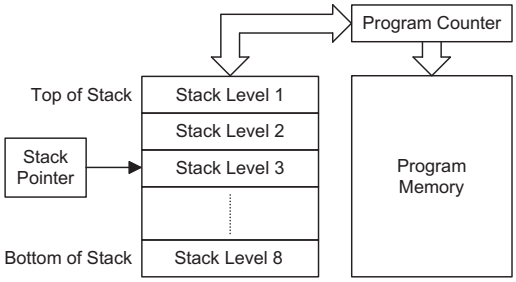
When executing instructions requiring jumps to non-consecutive addresses such as a jump instruction, a subroutine call, interrupt or reset, etc., the microcontroller manages program control by loading the required address into the Program Counter. For conditional skip instructions, once the condition 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.

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 writable 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.

The lower byte of the Program Counter is fully accessible under program control. Manipulating the PCL might cause program branching, so an extra cycle is needed to pre-fetch. Further information on the PCL register can be found in the Special Function Register section.

Stack

This is a special part of the memory which is used to save the contents of the Program Counter only. The stack has 8 levels and is neither part of the data nor part of the program space, and is neither readable nor writable. The activated level is indexed by the Stack Pointer, SP, and is neither readable nor writable. 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.



Mode	Program Counter												
	***12**	***11**	***10**	***9**	***8**	***7**	***6**	***5**	***4**	***3**	***2**	***1**	***0**
Initial Reset	0	0	0	0	0	0	0	0	0	0	0	0	0
External Interrupt	0	0	0	0	0	0	0	0	0	0	1	0	0
Timer 0 Overflow	0	0	0	0	0	0	0	0	0	1	0	0	0
Timer 1 Overflow	0	0	0	0	0	0	0	0	0	1	1	0	0
Timer 2 Overflow	0	0	0	0	0	0	0	0	1	0	0	0	0
Timer 3 Overflow	0	0	0	0	0	0	0	0	1	0	1	0	0
Skip	Program Counter + 2												
Loading PCL	*12	*11	*10	*9	*8	@7	@6	@5	@4	@3	@2	@1	@0
Jump, Call Branch	#12	#11	#10	#9	#8	#7	#6	#5	#4	#3	#2	#1	#0
Return from Subroutine	S12	S11	S10	S9	S8	S7	S6	S5	S4	S3	S2	S1	S0

Program Counter

Note: *12~*0: Program counter bits S12~S0: Stack register bits
#12~#0: Instruction code bits @7~@0: PCL bits
The program counter in the HT86B03 is only 12-bits wide therefore the *12 column in the table is not applicable.

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.

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

Program Memory

The Program Memory is the location where the user code or program is stored. By using the appropriate programming tools, this Program memory device offer users the flexibility to conveniently debug and develop their applications while also offering a means of field programming.

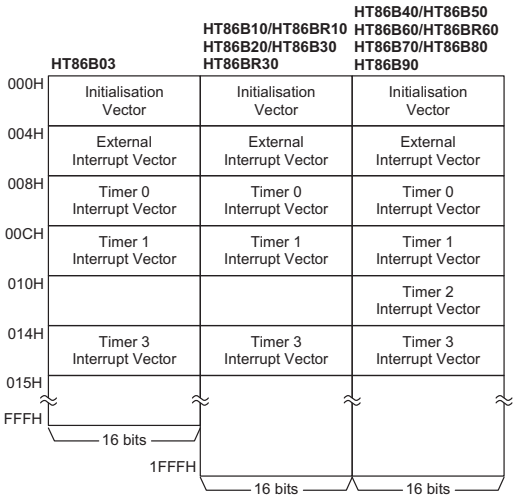
Organization

The program memory stores the program instructions that are to be executed. It also includes data, table and interrupt entries, addressed by the Program Counter along with the table pointer. The program memory size is 8192×16 bits. Certain locations in the program memory are reserved for special usage.

Special Vectors

Within the Program Memory, certain locations are reserved for special usage such as reset and interrupts.

- Location 000H
  This vector 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.
- Location 004H
  This vector is used by the external interrupt. If the external interrupt pin on the device goes low, the program will jump to this location and begin execution if the external interrupt is enabled and the stack is not full.
- Location 008H
  This internal vector is used by the 8-bit Timer 0. If a overflow occurs, the program will jump to this location and begin execution if the timer interrupt is enabled and the stack is not full.
- Location 00CH
  This internal vector is used by the 8-bit Timer1. If a overflow occurs, the program will jump to this location and begin execution if the timer interrupt is enabled and the stack is not full.
- Location 010H
  For the HT86B40, HT86B50, HT86B50, HT86B60, HT86BR60, HT86B70, HT86B80, HT86B90 devices, this internal vector is used by the 16-bit Timer2. If a overflow occurs, the program will jump to this location and begin execution if the timer interrupt is enabled and the stack is not full.
- Location 014H
  This internal vector is used by the 8-bit Timer3. If a overflow occurs, the program will jump to this location and begin execution if the timer interrupt is enabled and the stack is not full.



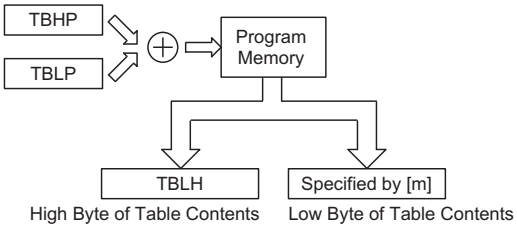
Program Memory Structure

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, table pointers are used to setup the address of the data that is to be accessed from the Program Memory. However, as some devices possess only a low byte table pointer and other devices possess both a high and low byte pointer it should be noted that depending upon which device is used, accessing look-up table data is implemented in slightly different ways.

For the devices, there are two Table Pointer Registers known as TBLP and TBHP in which the lower order and higher order address of the look-up data to be retrieved must be respectively first written. Unlike the other devices in which only the low address byte is defined using the TBLP register, the additional TBHP register allows the complete address of the look-up table to be defined and consequently allow table data from any address and any page to be directly accessed. For these devices, after setting up both the low and high byte table pointers, the table data can then be retrieved from any area of Program Memory using the "TABRDC [m]" instruction or from the last page of the Program Memory using the "TABRDL [m]" instruction. When either of these instructions are executed, the lower order table byte from the Program Memory will be transferred to the user defined Data Memory register [m] as specified in the instruction. The higher order table data byte from the Program Memory will be transferred to the TBLH special register. Any unused bits in this transferred higher order byte will be read as "0".

The following diagram illustrates the addressing/data flow of the look-up table for the devices:



Look-up Table

Table Program Example

The following example shows how the table pointer and table data is defined and retrieved from the devices. This example uses raw table data located in the last page which is stored there using the ORG statement. The value at this ORG statement is "1F00H" which refers to the start address of the last page within the Program Memory of the microcontroller. The table pointer is setup here to have an initial value of "06H". This will ensure that the first data read from the data table will be at the Program Memory address "1F06H" or 6 locations after the start of the last page. Note that the value for the table pointer is referenced to the first address of the present page if the "TABRDC [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 "TABRDL [m]" instruction is executed.

```
tempreg1    db      ?       ; temporary register #1
tempreg2    db      ?       ; temporary register #2
            :
            :

mov         a,06h           ; initialise table pointer - note that this address
                            ; is referenced

mov         tblp,a          ; to the last page or present page
            :
            :

tabrdl      tempreg1        ; transfers value in table referenced by table pointer
                            ; to tempregl
                            ; data at prog. memory address "1F06H" transferred to
                            ; tempreg1 and TBLH

dec         tblp            ; reduce value of table pointer by one

tabrdl      tempreg2        ; transfers value in table referenced by table pointer
                            ; to tempreg2
                            ; data at prog.memory address "1F05H" transferred to
                            ; tempreg2 and TBLH
                            ; in this example the data "1AH" is transferred to
                            ; tempreg1 and data "0FH" to register tempreg2
                            ; the value "00H" will be transferred to the high byte
                            ; register TBLH
            :
            :

org         1F00h           ; sets initial address of HT86B60 last page

dc          00Ah, 00Bh, 00Ch, 00Dh, 00Eh, 00Fh, 01Ah, 01Bh
            :
            :
```

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

Instruction	Table Location												
	*12	*11	*10	*9	*8	*7	*6	*5	*4	*3	*2	*1	*0
TABRDC [m]	P12	P11	P10	P9	P8	@7	@6	@5	@4	@3	@2	@1	@0
TABRDL [m]	1	1	1	1	1	@7	@6	@5	@4	@3	@2	@1	@0

Table Location

Note: *12~*0: Current Program ROM table
P12~P8: Write P12~P8 to TBHP pointer register
@7~@0: Write @7~@0 to TBLP pointer register
For the HT86B03, the table address location is 12-bits, that is from bit 0 to bit 11.

Data Memory

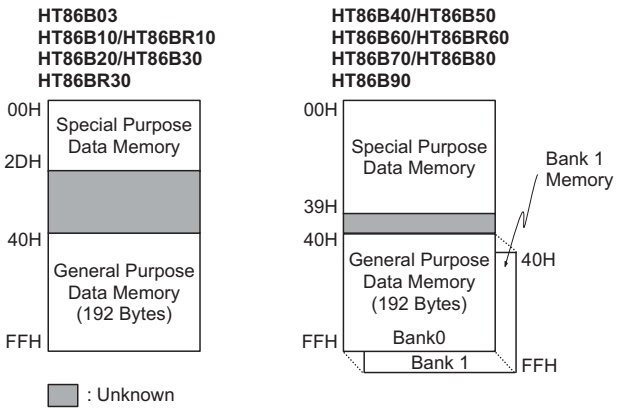
The Data Memory is a volatile area of 8-bit wide RAM internal memory and is the location where temporary information is stored. Divided into two sections, the first of these is an area of RAM where special function registers are located. These registers have fixed locations and are necessary for correct 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 RAM Data Memory is reserved for general purpose use. All locations within this area are read and write accessible under program control.

Organization

The Data Memory is subdivided into two banks, known as Bank 0 and Bank 1, all of which are implemented in 8-bit wide RAM. Most of the RAM Data Memory is located in Bank 0 which is also subdivided into two sections, the Special Purpose Data Memory and the General Purpose Data Memory. The length of these sections is dictated by the type of microcontroller chosen. The start address of the RAM Data Memory for all devices is the address ″00H″, and the last Data Memory address is ″FFH″. Registers which are common to all microcontrollers, such as ACC, PCL, etc., have the same Data Memory address.

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 program for both read and write operations. By using the



RAM Data Memory Structure – Bank 0, Bank1

Note: Most of the RAM Data Memory bits can be directly manipulated using the ″SET [m].i″ and ″CLR [m].i″ instructions with the exception of a few dedicated bits. The RAM Data Memory can also be accessed through the Memory Pointer registers MP0 and MP1.

″SET [m].i″ and ″CLR [m].i″ 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

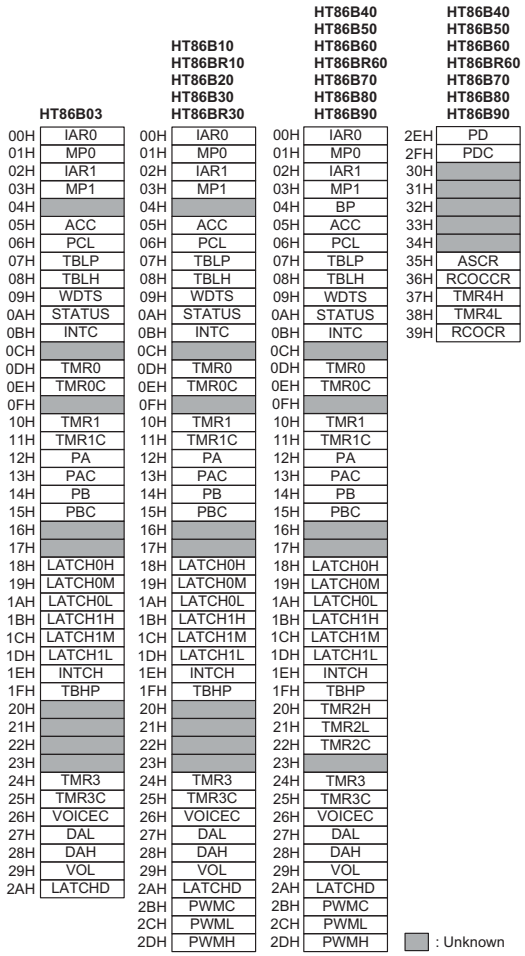
This area of Data Memory, is located in Bank 0, where registers, necessary for the correct operation of the microcontroller, are stored. Most of the registers are both readable and writable but some are protected and are readable only, the details of which are located under the relevant Special Function Register section. Note that for locations that are unused, any read instruction to these addresses will return the value ″00H″. Although the Special Purpose Data Memory registers are located in Bank 0, they will still be accessible even if the Bank Pointer has selected Bank 1.

Special Function Registers

To ensure successful operation of the microcontroller, certain internal registers are implemented in the RAM Data Memory area. These registers ensure correct operation of internal functions such as timers, interrupts, watchdog, etc., as well as external functions such as I/O data control. The location of these registers within the RAM Data Memory begins at the address ″00H″. Any unused Data Memory locations between these special function registers and the point where the General Purpose Memory begins is reserved for future expansion purposes, attempting to read data from these locations will return a value of ″00H″.

Indirect Addressing Register – 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 Pointer, MP0 or MP1. Acting as a pair, IAR0 and MP0 can together only access data from Bank 0, while the IAR1 and MP1 register pair can access data from both Bank 0 and Bank 1. As the Indirect Addressing Registers are not physically implemented, reading the Indirect Addressing Registers indirectly will return a result of ″00H″ and writing to the registers indirectly will result in no operation.



Special Purpose Data Memory Structure

Memory Pointer – MP0, MP1

For all devices, 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 only, while MP1 and IAR1 are used to access data from both Bank 0 and Bank 1.

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

```
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                ; increment memory pointer
             sdz block              ; check if last memory location has been cleared
             jmp loop

continue:
```

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

Bank Pointer – BP

The RAM Data Memory is divided into two Banks, known as Bank 0 and Bank 1. With the exception of the BP register, all of the Special Purpose Registers and General Purpose Registers are contained in Bank 0. If data in Bank 0 is to be accessed, then the BP register must be loaded with the value "00", while if data in Bank 1 is to be accessed, then the BP register must be loaded with the value ″01″.

Using Memory Pointer MP0 and Indirect Addressing Register IAR0 will always access data from Bank 0, irrespective of the value of the Bank Pointer.

The Data Memory is initialised to Bank 0 after a reset, except for the WDT time-out reset in the Power Down Mode, in which case, the Data Memory bank remains unaffected. It should be noted that Special Function Data Memory is not affected by the bank selection, which means that the Special Function Registers can be accessed from within either Bank 0 or Bank 1. Directly addressing the Data Memory will always result in Bank 0 being accessed irrespective of the value of the Bank Pointer.

Accumulator – ACC

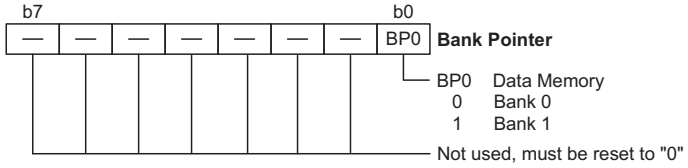
The Accumulator is central to the operation of any microcontroller and is closely related with operations carried out by the ALU. The Accumulator is the place where all intermediate results from the ALU are stored. Without the Accumulator it would be necessary to write the result of each calculation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

Program Counter Low Register – PCL

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

Look-up Table Registers – TBLP, TBLH

These two special function registers are used to control operation of the look-up table which is stored in the Program Memory. TBLP is the table pointer and indicates the location where the table data is located. Its value must be setup before any table read commands are executed. Its value can be changed, for example using the ″INC″ or ″DEC″ instructions, allowing for easy table data pointing and reading. TBLH is the location where the



Bank Pointer – BP

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.

Watchdog Timer Register – WDTS

The Watchdog feature of the microcontroller provides an automatic reset function giving the microcontroller a means of protection against spurious jumps to incorrect Program Memory addresses. To implement this, a timer is provided within the microcontroller which will issue a reset command when its value overflows. To provide variable Watchdog Timer reset times, the Watchdog Timer clock source can be divided by various division ratios, the value of which is set using the WDTS register. By writing directly to this register, the appropriate division ratio for the Watchdog Timer clock source can be setup. Note that only the lower 3 bits are used to set division ratios between 1 and 128.

Status Register – STATUS

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

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

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

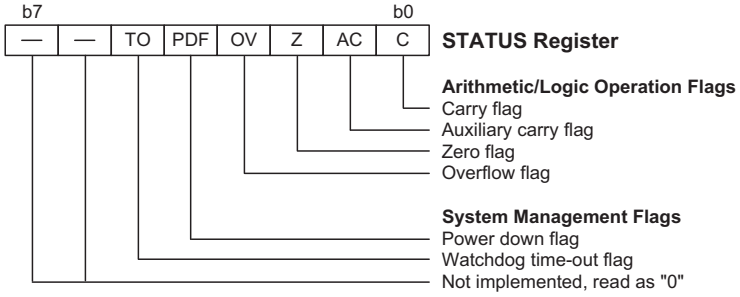
- **C** is set if an operation results in a carry during an addition operation or if a borrow does not take place during a subtraction operation; otherwise C is cleared. C is also affected by a rotate through carry instruction.
- **AC** is set if an operation results in a carry out of the low nibbles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

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

Interrupt Control Register – INTC, INTCH

Two 8-bit register, known as the INTC and INTCH registers, controls the operation of both external and internal timer interrupts. By setting various bits within these registers using standard bit manipulation instructions, the enable/disable function of the external and timer interrupts can be independently controlled. A master interrupt bit within this register, the EMI bit, acts like a global enable/disable and is used to set all of the interrupt enable bits on or off. This bit is cleared when an interrupt routine is entered to disable further interrupt and is set by executing the ″RETI″ instruction.

Note: In situations where other interrupts may require servicing within present interrupt service routines, the EMI bit can be manually set by the program after the present interrupt service routine has been entered.



Status Register

Timer Registers

Depending upon which device is selected, all devices contain three or four integrated Timers of either 8-bit or 16-bit size. All devices contain three 8-bit Timers whose associated registers are known as TMR0, TMR1 and TMR3, which is the location where the associated timer's 8-bit value is located. Their associated control registers, known as TMR0C, TMR1C and TMR3C, contain the setup information for these timers. Some devices also contain an additional 16-bit timer whose register pair name is known as TMR2L/TMR2H and is the location where the timer's 16-bit value is located. An associated control register, known as TMR2C, contains the setup information for this timer. Note that all timer registers can be directly written to in order to preload their contents with fixed data to allow different time intervals to be setup.

Input/Output Ports and Control Registers

Within the area of Special Function Registers, the I/O registers and their associated control registers play a prominent role. All I/O ports have a designated register correspondingly labeled as PA, PB, PD, etc. These labeled I/O registers are mapped to specific addresses within the Data Memory as shown in the Data Memory table, which are used to transfer the appropriate output or input data on that port. With each I/O port there is an associated control register labeled PAC, PBC, PDC, etc., also mapped to specific addresses with the Data Memory. The control register specifies which pins of that port are set as inputs and which are set as outputs. To setup a pin as an input, the corresponding bit of the control register must be set high, for an output it must be set low. During program initialisation, it is important to first setup the control registers to specify which pins are outputs and which are inputs before reading data from or writing data to the I/O ports. One flexible feature of these registers is the ability to directly program single bits using the "SET [m].i" and "CLR [m].i" instructions. The ability to change I/O pins from output to input and vice-versa by manipulating specific bits of the I/O control registers during normal program operation is a useful feature of these devices.

Voice ROM Data Address Latch Counter Registers

These are the LATCH0H/LATCH0M/LATCH0L, LATCH1H/LATCH1M/LATCH1L and the Voice ROM data registers. The voice ROM data address latch counter provides the handshaking between the microcontroller and the voice ROM, where the voice codes are stored. Eight bits of voice ROM data will be addressed by using the 22-bit address (except for the HT86B03 which has only 18-bits) latch counter, which is composed of LATCH0H/LATCH0M/LATCH0L or LATCH1H/LATCH1M/LATCH1L. After the 8-bit voice ROM data is addressed, several instruction cycles of at least 4us at least, will be required to latch the voice ROM data, after which the microcontroller can read the voice data from LATCHD.

Voice Control and Audio output Registers – VOICEC, DAL, DAH, VOL

The device includes a single 12-bit current type DAC function for driving an external 8Ω speaker through an external NPN transistor. The programmer must write the voice data to the DAL/DAH registers.

Pulse Width Modulator Registers – PWMC, PWML, PWMH

Each device contains a single 12-bit PWM function for driving an external 8Ω speaker. The programmer must write the voice data to PWML/PWMH register.

Analog Switch Registers – ASCR

Some devices, include 8 analog switch lines, which have an associated register, known as ASCR, for their setup and control.

External RC Oscillation Converter Registers – RCOCCR, RCOCR, TMR4L, TMR4H

For the HT86B40/HT86B50/HT86B60/HT86BR60/ HT86B70/ HT86B80/HT86B90 devices, which have two 16-bit programmable timers, the TMR4L and TMR4H registers are for one of the 16-bit timers. The RCOCCR and RCOCR registers are the control registers for the external RC oscillator.

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 options for all ports and wake-up options on certain pins, the user is provided with an I/O structure to meet the needs of a wide range of application possibilities.

Depending upon which device or package is chosen, the microcontroller range provides from 16 to 24 bidirectional input/output lines labeled with port names PA, PB, PD, etc. These I/O ports are mapped to the 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.

Pull-high Resistors

Many product applications require pull-high resistors for their switch inputs usually requiring the use of an external resistor. To eliminate the need for these external resistors, all I/O pins, when configured as an input have the capability of being connected to an internal pull-high resistor. These pull-high resistors are selectable via configuration options and are implemented using a weak PMOS transistor. Note that if the pull-high option is selected, then all I/O pins on that port will be connected to pull-high resistors, individual pins can be selected for pull-high resistor options.

Port A Wake-up

Each device has a HALT instruction enabling the microcontroller to enter a Power Down Mode and preserve 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. After a "HALT" instruction forces the microcontroller into entering a HALT condition, the processor will remain idle or in a low-power state until the logic condition of the selected wake-up pin on Port A changes from high to low. This function is especially suitable for applications that can be woken up via external switches. Note that each pin on Port A can be selected individually to have this wake-up feature.

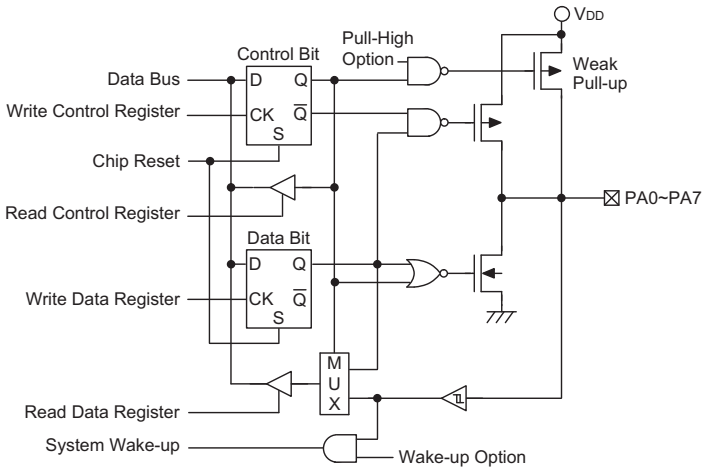
I/O Port Control Registers

Each I/O port has its own control register PAC, PBC, PDC, etc., to control the input/output configuration. With this control register, each CMOS output or input with or without pull-high resistor structures 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.

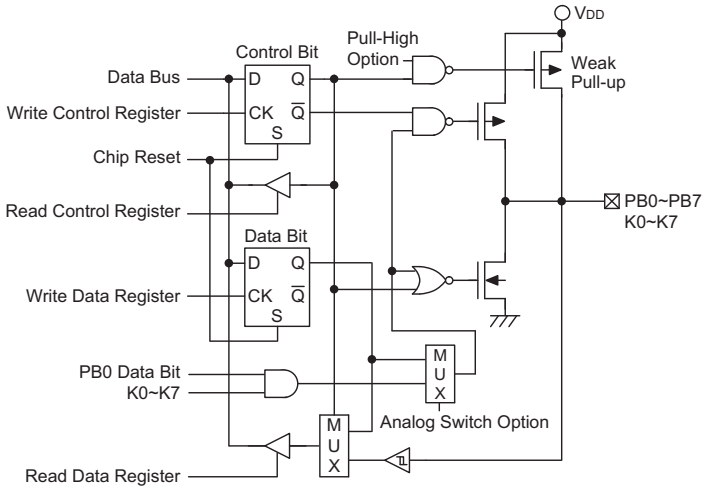
Pin-shared Functions

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 some pins, the chosen function of the multi-function I/O pins is set by configuration options while for others the function is set by application program control.

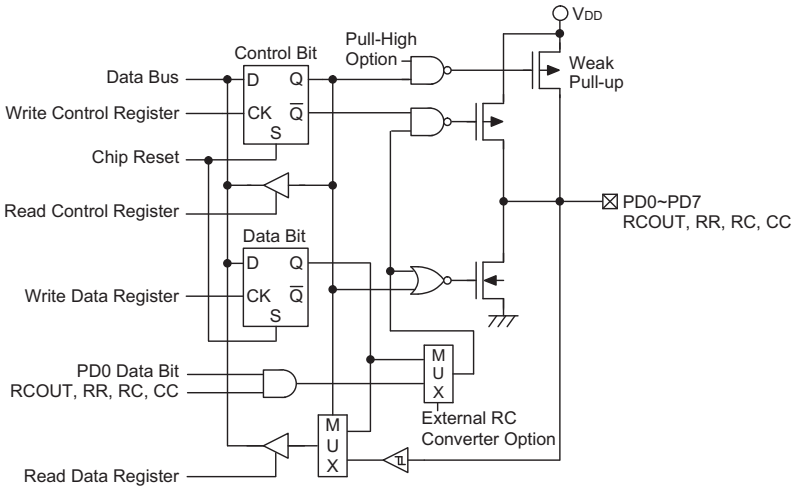
- Analog Switch
  For the HT86B40, HT86B50, HT86B60, HT86BR60, HT86B70, HT86B80 and HT86B90 devices, pins PB0~PB7 are pin-shared with analog switch pins K0 to K7. The choice of which function is used is selected using configuration options and remains fixed after the device is programmed.
- External RC Oscillator Converter
  For the HT86B40, HT86B50, HT86B60, HT86BR60, HT86B70, HT86B80 and HT86B90 devices, pins PD4~PD7 are pin-shared with external oscillator converter pins RCOUT, RR, RC and CC. The external RC oscillator converter function is selected via a configuration option and remains fixed after the device is programmed.
- I/O Pin Structures
  The following diagrams illustrate the I/O pin internal structures. As the exact logical construction of the I/O pin may differ from these drawings, they are supplied as a guide only to assist with the functional understanding of the I/O pins. Note also that the specified pins refer to the largest device package, therefore not all pins specified will exist on all devices.



PA Input/Output Port



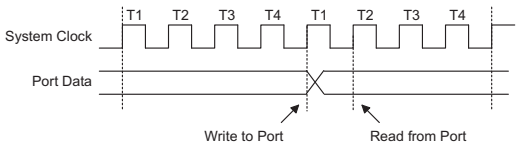
PB Input/Output Port



PD Input/Output Port

Programming Considerations

Within the user program, one of the first things to consider is port initialization. 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 options have been selected. If the port control registers, PAC, PBC, PDC, etc., 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, PA, PB, PD, etc., 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.



Read/Write Timing

Port A has the additional capability of providing wake-up functions. When the device is in the Power Down 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.

Timers

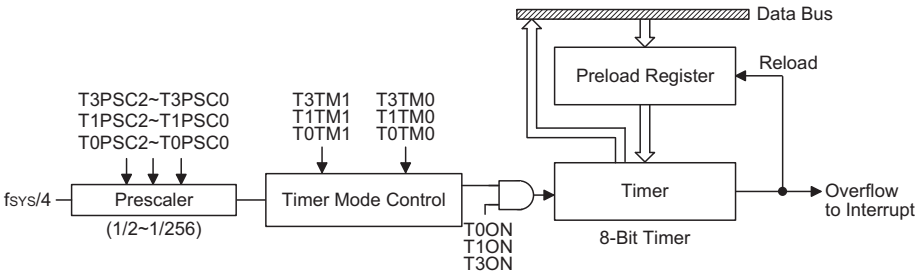
The provision of timers form an important part of any microcontroller, giving the designer a means of carrying out time related functions. The devices in the Voice Type MCU series contain either three or four count up timers of either 8 or 16-bit capacity depending upon which device is selected. The provision of an internal prescaler to the clock circuitry of some of the timer gives added range to the timer.

There is single type of register related to the Timer. The first is the register that contains the actual value of the timer and into which an initial value can be preloaded. Reading from this register retrieves the contents of the Timer. All devices can have the timer clock configured to come from the internal clock source. The accompanying table lists the associated timer register names.

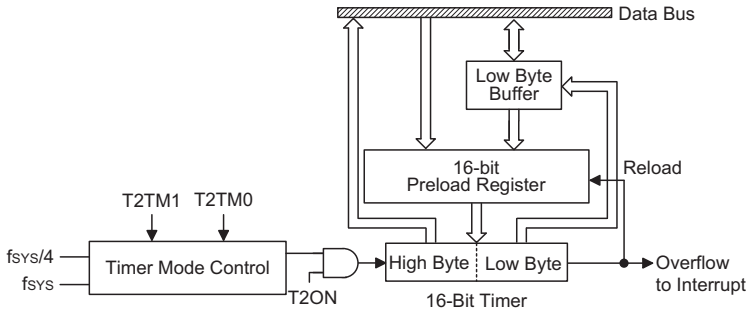
	HT86B03 HT86B10 HT86BR10 HT86B20 HT86B30 HT86BR30	**HT86B40 HT86B50 HT86B60 HT86BR60 HT86B70 HT86B80 HT86B90**
No. of 8-bit Timers	3	3
Timer Register Name	TMR0 TMR1 TMR3	TMR0 TMR1 TMR3
Timer Control Register	TMR0C TMR1C TMR3C	TMR0C TMR1C TMR3C
No. of 16-bit Timers	—	1
Timer Register Name	—	TMR2L TMR2H
Timer Control Register	—	TMR2C

Configuring the Timer Input Clock Source

The clock source for the 8-bit timers is the system clock divided by four while the 16-bit timer has a choice of either the system clock or the system clock divided by four. The 8-bit timer clock source is also first divided by the division ratio of which is conditioned by the three lower bits of the associated timer control register.



8-bit Timer Structure



16-bit Timer Structure – HT86B40/HT86B50/HT86B60/HT86BR60/HT86B70/HT86B80/HT86B90

Timer Registers – TMR0, TMR1, TMR2L/TMR2H, TMR3

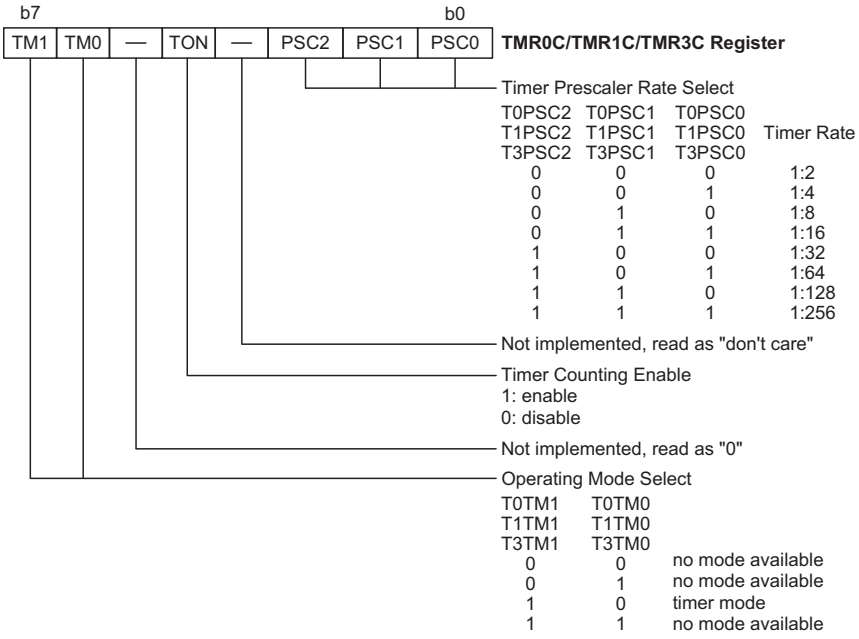
The timer registers are special function registers located in the special purpose Data Memory and is the place where the actual timer value is stored. All devices contain three 8-bit timers, whose registers are known as TMR0, TMR1 and TMR3. The HT86B40, HT86B50, HT86B60, HT86BR60, HT86B70, HT86B80 and HT86B90 devices also contain an additional single 16-bit timer, which has a pair of registers known as TMR2L and TMR2H. The value in the timer registers increases by one each time an internal clock pulse is received. The timer will count from the initial value loaded by the preload register to the full count of FFH for the 8-bit timer or FFFFH for the 16-bit timers at which point the timer overflows and an internal interrupt signal is generated. The timer value will then be reset with the initial preload register value and continue counting.

Note that to achieve a maximum full range count of FFH for the 8-bit timer or FFFFH for the 16-bit timers, the preload registers must first be cleared to all zeros. It should be noted that after power-on, the preload registers will be in an unknown condition. Note that if the Timer Counters are in an OFF condition and data is written to their preload registers, 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 the next time an overflow occurs. Note also that when the timer registers are read, the timer clock will be blocked to avoid errors, however, as this may result in certain timing errors, programmers must take this into account.

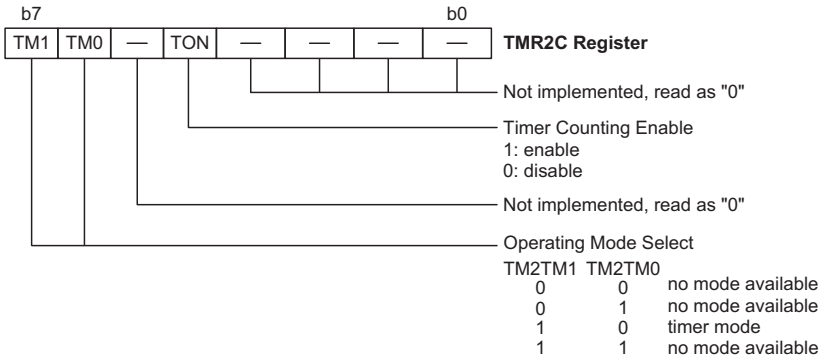
For devices which have an internal 16-bit Timer, and which therefore have both low byte and high byte timer registers, accessing these registers is carried out in a specific way. It must be noted that when using instructions to preload data into the low byte register, namely TMR2L, the data will only be placed in a low byte buffer and not directly into the low byte register. The actual transfer of the data into the low byte register is only carried out when a write to its associated high byte register, namely TMR2H, is executed. However, using instructions to preload data into the high byte timer register will result in the data being directly written to the high byte register. At the same time the data in the low byte buffer will be transferred into its associated low byte register. For this reason, when preloading data into the 16-bit timer registers, the low byte should be written first. It must also be noted that to read the contents of the low byte register, a read to the high byte register must first be executed to latch the contents of the low byte buffer into its associated low byte register. After this has been done, the low byte register can be read in the normal way. Note that reading the low byte timer register will only result in reading the previously latched contents of the low byte buffer and not the actual contents of the low byte timer register.

Timer Control Registers – TMR0C, TMR1C, TMR2C, TMR3C

Each timer has its respective timer control register, known as TMR0C, TMR1C, TMR2C and TMR3C. It is the timer control register together with their corresponding timer registers that control the full operation of the timers. Before the timers can be used, it is essential that the appropriate timer control register is fully programmed with the right data to ensure its correct operation, a process that is normally carried out during program initialization. Bits 7 and 6 of the Timer Control Register, which are known as the bit pair TM1/TM0 respectively, must be set to the required logic levels. The timer-on bit, which is bit 4 of the Timer Control Register and known as TON, depending upon which timer is used, provides the basic on/off control of the respective timer. setting the bit high allows the timer to run, clearing the bit stops the timer. For the 8-bit timers, which have prescalers, bits 0~2 of the Timer Control Register determine the division ratio of the input clock prescaler.



Timer Control Register – All Devices



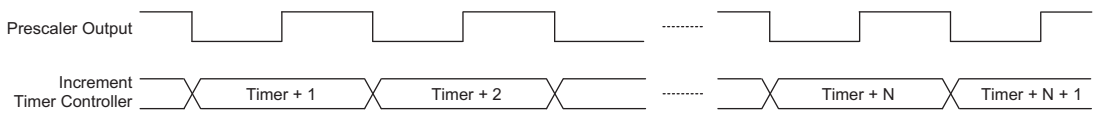
Timer Control Register – HT86B40/HT86B50/HT86B60/HT86BR60/HT86B70/HT86B80/HT86B90

Configuring the Timer

The Timer is used to measure fixed time intervals, providing an internal interrupt signal each time the Timer overflows. To do this the Operating Mode Select bit pair in the Timer Control Register must be set to the correct value as shown.

Control Register Operating Mode Select Bits	Bit7	Bit6
	1	0

The internal clock, f_{SYS}, is used as the Timer clock. However, this clock source is further divided by a prescaler, the value of which is determined by the Prescaler Rate Select bits, which are bits 0~2 in the Timer Control Register. After the other bits in the Timer Control Register have been setup, the enable bit, which is bit 4 of the Timer Control Register, can be set high to enable the Timer to run. Each time an internal clock cycle occurs, the Timer increments by one. When it is full and overflows, an interrupt signal is generated and the Timer will reload the value already loaded into the preload register and continue counting. The interrupt can be disabled by ensuring that the Timer Interrupt Enable bit in the Interrupt Control Register, INTC, is reset to zero.



Timer Mode Timing Diagram

Prescaler

All of the 8-bit timers possess a prescaler. Bits 0~2 of their associated timer control register, define the pre-scaling stages of the internal clock source of the Timer. The Timer overflow signal can be used to generate signals for the Timer interrupt.

Programming Considerations

The internal system clock is used as the timer clock source and is therefore synchronized with the overall operation of the microcontroller. In this mode, when the appropriate timer register is full, the microcontroller will generate an internal interrupt signal directing the program flow to the respective internal interrupt vector.

When the Timer is read, the clock is blocked to avoid errors, however as this may result in a counting error, this should be taken into account by the programmer. Care must be taken to ensure that the timers are properly initialized before using them for the first time. The associated timer enable bits in the interrupt control register must be properly set otherwise the internal interrupt associated with the timer will remain inactive. The edge select, timer mode and clock source control bits in timer control register must also be correctly set to ensure the timer is properly configured for the required application. It is also important to ensure that an initial value is first loaded into the timer registers before the timer is switched on; this is because after power-on the initial values of the timer registers are unknown. After the timer has been initialized the timer can be turned on and off by controlling the enable bit in the timer control register.

Timer Program Example

The following example program section is based on the HT86B40, HT86B50, HT86B60, HT86BR60, HT86B70, HT86B80 and HT86B90 devices, which contain a single internal 16-bit timer. Programming the timer for other devices is conducted in a very similar way. The program shows how the timer registers are setup along with how the interrupts are enabled and managed. Points to note in the example are how, for the 16-bit timer, the low byte must be written first, this is because the 16-bit data will only be written into the actual timer register when the high byte is loaded. Also note how the timer is turned on by setting bit 4 of the respective timer control register. The timer can be turned off in a similar way by clearing the same bit. This example program sets the timer to be in the timer mode which uses the internal system clock as their clock source.

```
#include HT86B40.inc
jmp begin
:
org 04h             ; external interrupt vectors
reti
org 08h
reti
org 0Ch
reti
org 10h             ; timer 2 interrupt vector
jmp tmr2int         ; jump here when timer 2 overflows
org 14h
reti
:
; internal timer 2 interrupt routine
tmr2int:
:
; timer 2 main program placed here
:
reti
:
begin:
; setup timer 2 registers
mov a,09bh          ; setup timer 2 low byte
mov tmr2l,a         ; low byte must be setup before high byte
mov a,0e8h          ; setup timer 2 high byte
mov tmr2h,a         ; setup timer 2 high byte
mov a,090h          ; setup timer 2 control register
mov tmr2c,a         ; setup timer mode
; setup interrupt register
mov a,01h           ; enable master interrupt
mov intc,a
mov a,01h           ; enable timer 2 interrupt
mov intch,a
:
```

Interrupts

Interrupts are an important part of any microcontroller system. When an external event or an internal function such as a Timer requires microcontroller attention, their corresponding interrupt will enforce a temporary suspension of the main program allowing the microcontroller to direct attention to their respective needs. Each device contains a single external interrupt and three or four internal timer interrupt functions. The external interrupt is controlled by the action of the external $\overline{INT}$ pin, while the internal interrupt is controlled by the relevant Timer overflow.

Interrupt Register

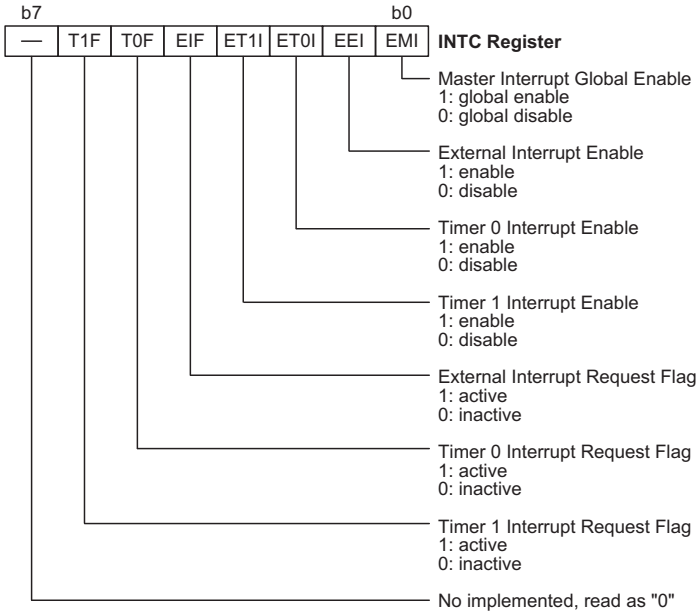
Overall interrupt control, which means interrupt enabling and flag setting, is controlled using two registers, known as INTC and INTCH, which are located in the Data Memory. By controlling the appropriate enable bits in these registers each individual interrupt can be enabled or disabled. Also when an interrupt occurs, the corresponding request flag will be set by the microcontroller. The global enable flag if cleared to zero will disable all interrupts.

Interrupt Operation

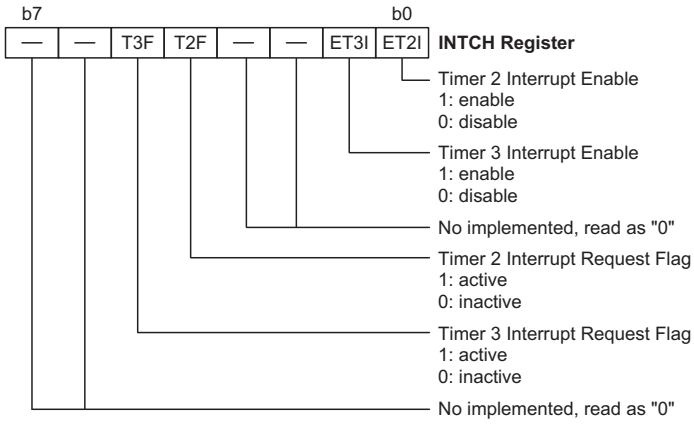
A timer overflow or the external interrupt line being pulled low will all generate an interrupt request by setting their corresponding request flag, if their appropriate interrupt enable bit is set. When this happens, the Program Counter, which stores the address of the next instruction to be executed, 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 statement which will take program execution 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 statement, 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 diagram with their order of priority.

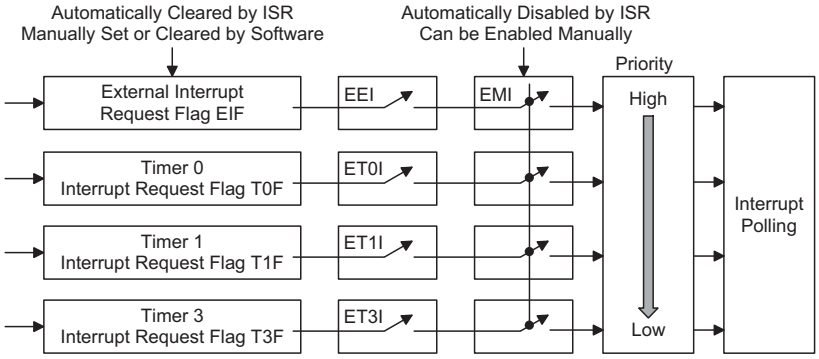
Once an interrupt subroutine is serviced, all the other interrupts will be blocked, as the EMI bit will be cleared automatically. This will prevent any further interrupt nesting from occurring. However, if other interrupt requests occur during this interval, although the interrupt will not be immediately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



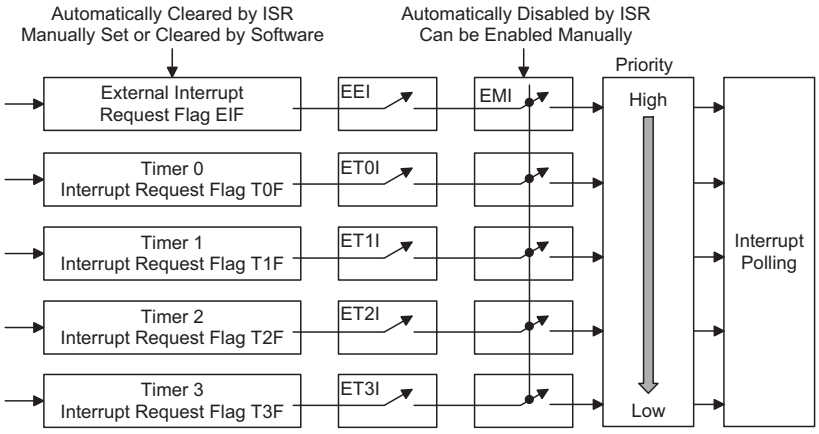
Interrupt Control Register



INTCH Register



Interrupt Structure – HT86B03/HT86B10/HT86BR10/HT86B20/HT86B30/HT86BR30



Interrupt Structure – HT86B40/HT86B50/HT86B60/HT86BR60/HT86B70/HT86B80/HT86B90

Interrupt Priority

Interrupts, occurring in the interval between the rising edges of two consecutive T2 pulses, will be serviced on the latter of the two T2 pulses, if the corresponding interrupts are enabled. In case of simultaneous requests, the accompanying table shows the priority that is applied.

Interrupt Source	Interrupt Vector	HT86B03/HT86B10 HT86BR10/HT86B20 HT86B30/HT86BR30 Priority	HT86B40/HT86B50/HT86B60 HT86BR60/HT86B70/HT86B80 HT86B90 Priority
External Interrupt	04H	1	1
Timer 0 Overflow	08H	2	2
Timer 1 Overflow	0CH	3	3
Timer 2 Overflow	10H	—	4
Timer 3 Overflow	14H	4	5

In cases where both external and timer interrupts are enabled and where an external and timer interrupt occur simultaneously, the external interrupt will always have priority and will therefore be serviced first. Suitable masking of the individual interrupts using the INTC and INTCH registers can prevent simultaneous occurrences.

External Interrupt

Each device contains a single external interrupt function controlled by the external pin, $\overline{INT}$. For an external interrupt to occur, the corresponding external interrupt enable bit must be first set. This is bit 1 of the INTC register and known as EEI. An external interrupt is triggered by an external edge transition on the external interrupt pin $\overline{INT}$, after which the related interrupt request flag, EIF, which is bit 4 of INTC, will be set. A configuration option exists for the external interrupt pin to determine the type of external edge transition which will trigger an external interrupt. There are two options available, a low going edge or both high and low going edges. When the master interrupt and external interrupt bits are enabled, the stack is not full and an active edge transition, as setup in the configuration options, occurs on the $\overline{INT}$ pin, a subroutine call to the corresponding external interrupt vector, which is located at 04H, will occur. After entering the interrupt execution routine, the corresponding interrupt request flag, EIF, will be reset and the EMI bit will be cleared to disable other interrupts.

Timer Interrupt

For a timer generated interrupt to occur, the corresponding timer interrupt enable bit must be first set. Each device contains three 8-bit timers whose corresponding interrupt enable bits are known as ET0I, ET1I and ET3I and are located in the INTC and INTCH registers. Each timer also has a corresponding timer interrupt request flag, which are known as T0F, T1F and T3F, also located in the INTC and INTCH registers. Some devices also contain a 16-bit timer, which has a corresponding timer interrupt enable bit, ET2I, and a corresponding timer request flag, T2F, which are contained in the INTCH register. When the master interrupt and corresponding timer interrupt enable bits are enabled, the stack is not full, and when the corresponding timer overflows a subroutine call to the corresponding timer interrupt vector will occur. The corresponding Program Memory vector locations for Timer 0, Timer1, Timer 2 and Timer 3 are 08H, 0CH, 10H and 14H. After entering the interrupt execution routine, the corresponding interrupt request flags, T0F, T1F, T2F or T3F will be reset and the EMI bit will be cleared to disable other interrupts.

Programming Considerations

By disabling the 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 INTC or INTCH register until the corresponding interrupt is serviced or until the request flag is cleared by a software instruction.

It is recommended that programs do not use the "CALL subroutine" instruction within the interrupt subroutine. Interrupts often occur in an unpredictable manner or need to be serviced immediately in some applications. 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.

All of these interrupts have the capability of waking up the processor when in the Power Down Mode. Only the Program Counter is pushed onto the stack. If the contents of the register or status register are altered by the interrupt service program, which may corrupt the desired control sequence, then the contents should be saved in advance.

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.

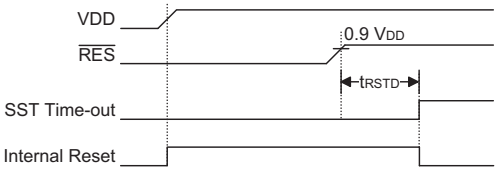
In addition to the power-on reset, situations may arise where it is necessary to forcefully apply a reset condition when the microcontroller is running. One example of this is where after power has been applied and the microcontroller is already running, the $\overline{\text{RES}}$ line is forcefully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

Another reset exists in the form of a Low Voltage Reset, LVR, where a full reset, similar to the $\overline{\text{RES}}$ reset is implemented in situations where the power supply voltage falls below a certain threshold.

Reset Functions

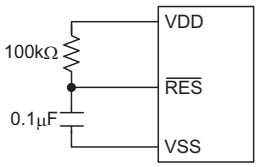
There are five ways in which a microcontroller reset can occur, through events occurring both internally and externally:

- Power-on Reset
  The most fundamental and unavoidable reset is the one that occurs after power is first applied to the 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.
  Although the microcontroller has an internal RC reset function, if the VDD power supply rise time is not fast enough or does not stabilise quickly at power-on, the internal reset function may be incapable of providing proper reset operation. For this reason it is recommended that an external RC network is connected to the $\overline{\text{RES}}$ pin, whose additional time delay will ensure that the $\overline{\text{RES}}$ pin remains low for an extended period to allow the power supply to stabilise. During this time delay, normal operation of the microcontroller will be inhibited. After the $\overline{\text{RES}}$ line reaches a certain voltage value, the reset delay time t_{RSTD} is invoked to provide an extra delay time after which the microcontroller will begin normal operation. The abbreviation SST in the figures stands for System Start-up Timer.



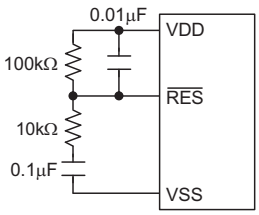
Power-On Reset Timing Chart

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



Basic Reset Circuit

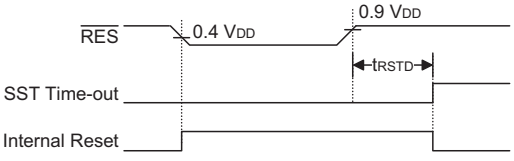
For applications that operate within an environment where more noise is present the Enhanced Reset Circuit shown is recommended.



Enhanced Reset Circuit

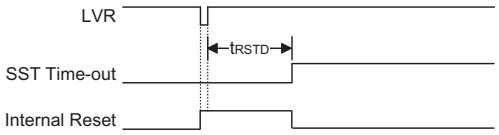
More information regarding external reset circuits is located in Application Note HA0075E on the Holtek website.

- $\overline{\text{RES}}$ Pin Reset
  This type of reset occurs when the microcontroller is already running and the $\overline{\text{RES}}$ pin is forcefully pulled low by external hardware such as an external switch. In this case as in the case of other reset, the Program Counter will reset to zero and program execution initiated from this point.



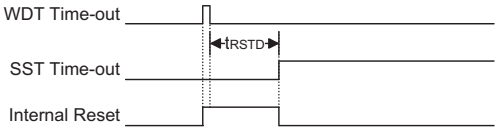
$\overline{\text{RES}}$ Reset Timing Chart

- Low Voltage Reset – LVR
  The microcontroller contains a low voltage reset circuit in order to monitor the supply voltage of the device, which is selected via a configuration option. If the supply voltage of the device drops to within a range of 0.9V~V_{LVR} such as might occur when changing the battery, the LVR will automatically reset the device internally. The LVR includes the following specifications: For a valid LVR signal, a low voltage, i.e., a voltage in the range between 0.9V~V_{LVR} must exist for greater than the value t_{LVR} specified in the A.C. characteristics. If the low voltage state does not exceed 1ms, the LVR will ignore it and will not perform a reset function.



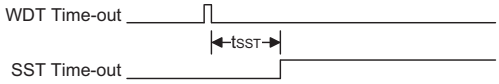
Low Voltage Reset Timing Chart

- Watchdog Time-out Reset during Normal Operation
  The Watchdog time-out Reset during normal operation is the same as a hardware $\overline{RES}$ pin 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 Power Down
  The Watchdog time-out Reset during Power Down 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 A.C. Characteristics for t_{SST} details.



WDT Time-out Reset during Power Down 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 Power Down function or Watchdog Timer. The reset flags are shown in the table:

TO	PDF	RESET Conditions
0	0	$\overline{RES}$ reset during power-on
u	u	$\overline{RES}$ or LVR reset during normal operation
1	u	WDT time-out reset during normal operation
1	1	WDT time-out reset during Power Down

Note: ″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	Clear after reset, WDT begins counting
Timer	All Timer will be turned off
Prescaler	The Timer Prescaler will be cleared
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.

HT86B03

Register	Reset (Power-on)	WDT Time-out (Normal Operation)	$\overline{\text{RES}}$ Reset (Normal Operation)	$\overline{\text{RES}}$ Reset (HALT)	WDT Time-out from HALT
MP0	0000 0000	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu
MP1	0000 0000	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu
ACC	0000 0000	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu
PCL	0000 0000	0000 0000	0000 0000	0000 0000	0000 0000
TBLP	0000 0000	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu
TBLH	0000 0000	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu
WDTS	0000 0111	0000 0111	0000 0111	0000 0111	uuuu uuuu
STATUS	--00 0000	--1u uuuu	--uu uuuu	--01 uuuu	--11 uuuu
INTC	-000 0000	-000 0000	-000 0000	-000 0000	-uuu uuuu
TMR0	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu
TMR0C	00-0 1000	00-0 1000	00-0 1000	00-0 1000	uu-u uuuu
TMR1	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu
TMR1C	00-0 1000	00-0 1000	00-0 1000	00-0 1000	uu-u uuuu
PA	1111 1111	1111 1111	1111 1111	1111 1111	uuuu uuuu
PAC	1111 1111	1111 1111	1111 1111	1111 1111	uuuu uuuu
PB	1111 ----	1111 ----	1111 ----	1111 ----	uuuu ----
PBC	1111 ----	1111 ----	1111 ----	1111 ----	uuuu ----
TMR3	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu
TMR3C	00-0 1000	00-0 1000	00-0 1000	00-0 1000	uu-u uuuu
INTCH	--0- --0-	--0- --0-	--0- --0-	--0- --0-	--u- --u-
TBHP	0000 0000	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu
DAL	0000 ----	uuuu ----	uuuu ----	uuuu ----	uuuu ----
DAH	0000 0000	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu
VOL	000- 0000	000- ----	000- ----	000- ----	uuu- ----
VOICEC	---0 -00-	---0 -00-	---0 -00-	---0 -00-	---u -uu-
LATCH0H	0000 0000	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu
LATCH0M	0000 0000	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu
LATCH0L	0000 0000	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu
LATCH1H	0000 0000	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu
LATCH1M	0000 0000	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu
LATCH1L	0000 0000	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu
LATCHD	0000 0000	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu

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

HT86B10/HT86BR10/HT86B20/HT86B30/HT86BR30

Register	Reset (Power-on)	WDT Time-out (Normal Operation)	$\overline{RES}$ Reset (Normal Operation)	$\overline{RES}$ Reset (HALT)	WDT Time-out from HALT
MP0	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu
MP1	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu
ACC	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu
PCL	0000 0000	0000 0000	0000 0000	0000 0000	0000 0000
TBLP	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu
TBLH	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu
WDTS	0000 0111	0000 0111	0000 0111	0000 0111	uuuu uuuu
STATUS	--00 xxxx	--1u uuuu	--uu uuuu	--01 uuuu	--11 uuuu
INTC	-000 0000	-000 0000	-000 0000	-000 0000	-uuu uuuu
TMR0	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu
TMR0C	00-0 1000	00-0 1000	00-0 1000	00-0 1000	uu-u uuuu
TMR1	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu
TMR1C	00-0 1000	00-0 1000	00-0 1000	00-0 1000	uu-u uuuu
PA	1111 1111	1111 1111	1111 1111	1111 1111	uuuu uuuu
PAC	1111 1111	1111 1111	1111 1111	1111 1111	uuuu uuuu
PB	1111 1111	1111 1111	1111 1111	1111 1111	uuuu uuuu
PBC	1111 1111	1111 1111	1111 1111	1111 1111	uuuu uuuu
TMR3	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu
TMR3C	00-0 1000	00-0 1000	00-0 1000	00-0 1000	uu-u uuuu
INTCH	--0- --0-	--0- --0-	--0- --0-	--0- --0-	--u- --u-
TBHP	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu
DAL	xxxx ----	uuuu ----	uuuu ----	uuuu ----	uuuu ----
DAH	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu
VOL	xxx- xxxx	uuu- uuuu	uuu- uuuu	uuu- uuuu	uuu- uuuu
VOICEC	---0 -00-	---0 -00-	---0 -00-	---0 -00-	---u -uu-
LATCH0H	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu
LATCH0M	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu
LATCH0L	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu
LATCH1H	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu
LATCH1M	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu
LATCH1L	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu
LATCHD	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu
PWMC	0--- 0--0	0--- 0--0	0--- 0--0	0--- 0--0	u--- u--u
PWML	xxxx ----	uuuu ----	uuuu ----	uuuu ----	uuuu ----
PWMH	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu

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

HT86B40/HT86B50/HT86B60/HT86BR60/HT86B70/HT86B80/HT86B90

Register	Reset (Power-on)	WDT Time-out (Normal Operation)	$\overline{\text{RES}}$ Reset (Normal Operation)	$\overline{\text{RES}}$ Reset (HALT)	WDT Time-out from HALT
MP0	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu
MP1	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu
BP	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu
ACC	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu
PCL	0000 0000	0000 0000	0000 0000	0000 0000	0000 0000
TBLP	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu
TBLH	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu
WDTS	0000 0111	0000 0111	0000 0111	0000 0111	uuuu uuuu
STATUS	--00 xxxx	--1u uuuu	--uu uuuu	--01 uuuu	--11 uuuu
INTC	-000 0000	-000 0000	-000 0000	-000 0000	-uuu uuuu
TMR0	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu
TMR0C	00-0 1000	00-0 1000	00-0 1000	00-0 1000	uu-u uuuu
TMR1	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu
TMR1C	00-0 1000	00-0 1000	00-0 1000	00-0 1000	uu-u uuuu
PA	1111 1111	1111 1111	1111 1111	1111 1111	uuuu uuuu
PAC	1111 1111	1111 1111	1111 1111	1111 1111	uuuu uuuu
PB	1111 1111	1111 1111	1111 1111	1111 1111	uuuu uuuu
PBC	1111 1111	1111 1111	1111 1111	1111 1111	uuuu uuuu
PD	1111 1111	1111 1111	1111 1111	1111 1111	uuuu uuuu
PDC	1111 1111	1111 1111	1111 1111	1111 1111	uuuu uuuu
TMR2H	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu
TMR2L	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu
TMR2C	00-0 ----	00-0 ----	00-0 ----	00-0 ----	uu-u ----
TMR3	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu
TMR3C	00-0 1000	00-0 1000	00-0 1000	00-0 1000	uu-u uuuu
INTCH	--00 --00	--00 --00	--00 --00	--00 --00	--uu --uu
TBHP	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu
DAL	xxxx ----	uuuu ----	uuuu ----	uuuu ----	uuuu ----
DAH	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu
VOL	xxx- xxxx	uuu- uuuu	uuu- uuuu	uuu- uuuu	uuu- uuuu
VOICEC	---0 -00-	---0 -00-	---0 -00-	---0 -00-	---u -uu-
LATCH0H	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu
LATCH0M	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu
LATCH0L	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu
LATCH1H	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu
LATCH1M	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu
LATCH1L	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu
LATCHD	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu

Register	Reset (Power-on)	WDT Time-out (Normal Operation)	$\overline{RES}$ Reset (Normal Operation)	$\overline{RES}$ Reset (HALT)	WDT Time-out from HALT
PWMC	0 – – – 0 – – 0	0 – – – 0 – – 0	0 – – – 0 – – 0	0 – – – 0 – – 0	u – – – u – – u
PWML	x x x x – – – –	u u u u – – – –	u u u u – – – –	u u u u – – – –	u u u u – – – –
PWMH	x x x x x x x x	u u u u u u u u	u u u u u u u u	u u u u u u u u	u u u u u u u u
ASCR	– – – – 1 1 1 1	– – – – 1 1 1 1	– – – – 1 1 1 1	– – – – 1 1 1 1	– – – – u u u u
RCOCCR	0 0 1 0 – – – –	0 0 1 0 – – – –	0 0 1 0 – – – –	0 0 1 0 – – – –	u u u u – – – –
TMR4H	x x x x x x x x	x x x x x x x x	x x x x x x x x	x x x x x x x x	u u u u u u u u
TMR4L	x x x x x x x x	x x x x x x x x	x x x x x x x x	x x x x x x x x	u u u u u u u u
RCOCR	1 x x x – – 0 0	1 x x x – – 0 0	1 x x x – – 0 0	1 x x x – – 0 0	u u u u – – u u

Note: "u" stands for unchanged
"x" stands for unknown
"–" stands for undefined

Oscillator

Various oscillator options offer the user a wide range of functions according to their various application requirements. Two types of system clocks can be selected while various clock source options for the Watchdog Timer are provided for maximum flexibility. All oscillator options are selected through the configuration options.

The two methods of generating the system clock are:

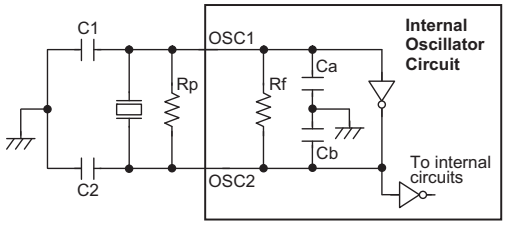
- External crystal/resonator oscillator
- External RC oscillator

One of these two methods must be selected using the configuration options.

More information regarding the oscillator is located in Application Note HA0075E on the Holtek website.

External Crystal/Resonator Oscillator

The simple connection of a crystal across OSC1 and OSC2 will create the necessary phase shift and feedback for oscillation, and will normally not require external capacitors. However, for some crystals and most resonator types, to ensure oscillation and accurate frequency generation, it may be necessary to add two small value external capacitors, C1 and C2. The exact values of C1 and C2 should be selected in consultation



Note: 1. Rp is normally not required.
2. Although not shown OSC1/OSC2 pins have a parasitic capacitance of around 7pF.

Crystal/Resonator Oscillator

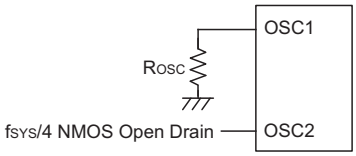
with the crystal or resonator manufacturer's specification. The external parallel feedback resistor, Rp, is normally not required but in some cases may be needed to assist with oscillation start up.

Internal Ca, Cb, Rf Typical Values @ 5V, 25°C		
Ca	**Cb**	**Rf**
11~13pF	13~15pF	800kΩ

Oscillator Internal Component Values

External RC Oscillator

Using the external system RC oscillator requires that a resistorco. The mask MCU value between 60kΩ and 130kΩ, the OTP MCU value between 150kΩ and 300kΩ. They connected between OSC1 and VSS. The generated system clock divided by 4 will be provided on OSC2 as an output which can be used for external synchronization purposes. Note that as the OSC2 output is an NMOS open-drain type, a pull high resistor should be connected if it to be used to monitor the internal frequency. Although this is a cost effective oscillator configuration, the oscillation frequency can vary with VDD, temperature and process variations and is therefore not suitable for applications where timing is critical or where accurate oscillator frequencies are required. Note that it is the only microcontroller internal circuitry together with the external resistor, that determine the frequency of the oscillator. The external capacitor shown on the diagram does not influence the frequency of oscillation.



External RC Oscillator

Watchdog Timer Oscillator

The WDT oscillator is a fully self-contained free running on-chip RC oscillator with a typical period of 65μs at 5V requiring no external components. When the device enters the Power Down Mode, the system clock will stop running but the WDT oscillator continues to free-run and to keep the watchdog active. However, to preserve power in certain applications the WDT oscillator can be disabled via a configuration option.

Power Down Mode and Wake-up

Power Down Mode

All of the Holtek microcontrollers have the ability to enter a Power Down Mode, also known as the HALT Mode or Sleep Mode. When the device enters this mode, the normal operating current, will be reduced to an extremely low standby current level. This occurs because when the device enters the Power Down Mode, the system oscillator is stopped which reduces the power consumption to extremely low levels, however, as the device maintains its present internal condition, it can be woken up at a later stage and continue running, without requiring a full reset. This feature is extremely important in application areas where the MCU must have its power supply constantly maintained to keep the device in a known condition but where the power supply capacity is limited such as in battery applications.

Entering the Power Down Mode

There is only one way for the device to enter the Power Down Mode and that is to execute the "HALT" instruction in the application program. When this instruction is executed, the following will occur:

- The system oscillator will stop running and the application program will stop at the "HALT" instruction.
- The Data Memory contents and registers will maintain their present condition.
- The WDT will be cleared and resume counting if the WDT clock source is selected to come from the WDT oscillator. The WDT will stop if its clock source originates from the system clock.
- The I/O ports will maintain their present condition.
- In the status register, the Power Down flag, PDF, will be set and the Watchdog time-out flag, TO, will be cleared.

Standby Current Considerations

As the main reason for entering the Power Down Mode is to keep the current consumption of the MCU to as low a value as possible, perhaps only in the order of several micro-amps, there are other considerations which must also be taken into account by the circuit designer if the power consumption is to be minimized. 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. Care must also be taken with the loads, which are connected to I/Os, 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 configuration options have enabled the Watchdog Timer internal oscillator.

Wake-up

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

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

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

Each pin on Port A can be setup via an individual configuration option to permit a negative transition on the pin to wake-up the system. When a Port 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 to "1" before entering the Power Down Mode, the wake-up function of the related interrupt will be disabled.

No matter what the source of the wake-up event is, once a wake-up situation occurs, a time period equal to 1024 system clock periods will be required before normal system operation resumes. However, if the wake-up has originated due to an interrupt, the actual interrupt subroutine execution will be delayed by an additional one or more cycles. If the wake-up results in the execution of the next instruction following the "HALT" instruction, this will be executed immediately after the 1024 system clock period delay has ended.

Watchdog Timer

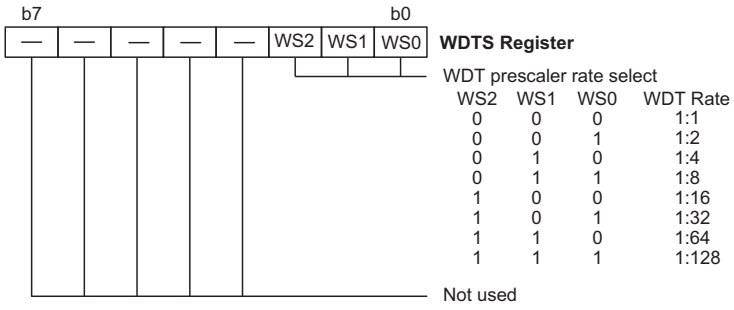
The Watchdog Timer is provided to prevent program malfunctions or sequences from jumping to unknown locations, due to certain uncontrollable external events such as electrical noise. It operates by providing a device reset when the WDT counter overflows. The WDT clock is supplied by one of two sources selected by configuration option: its own self-contained dedicated internal WDT oscillator, or the instruction clock which is the system clock divided by 4. Note that if the WDT configuration option has been disabled, then any instruction relating to its operation will result in no operation.

The internal WDT oscillator has an approximate period of 65μs at a supply voltage of 5V. If selected, it is first divided by 256 via an 8-stage counter to give a nominal period of 17ms. Note that this period can vary with VDD, temperature and process variations. For longer WDT time-out periods the WDT prescaler can be utilized. By writing the required value to bits 0, 1 and 2 of the WDTS register, known as WS0, WS1 and WS2, longer time-out periods can be achieved. With WS0, WS1 and WS2 all equal to 1, the division ratio is 1:128 which gives a maximum time-out period of about 2.1s.

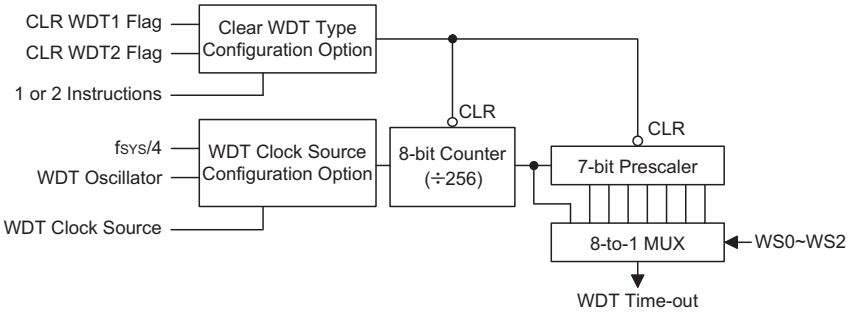
A configuration option can select the instruction clock, which is the system clock divided by 4, as the WDT clock source instead of the internal WDT oscillator. If the instruction clock is used as the clock source, it must be noted that when the system enters the Power Down Mode, as the system clock is stopped, then the WDT clock source will also be stopped. Therefore the WDT will lose its protecting purposes. In such cases the system cannot be restarted by the WDT and can only be restarted using external signals. For systems that operate in noisy environments, using the internal WDT oscillator is therefore the recommended choice.

Under normal program operation, a WDT time-out will initialise a device reset and set the status bit TO. However, if the system is in the Power Down Mode, when a WDT time-out occurs, only the Program Counter and Stack Pointer will be reset. Three methods can be adopted to clear the contents of the WDT and the WDT prescaler. The first is an external hardware reset, which means a low level on the $\overline{RES}$ pin, the second is using the watchdog software instructions and the third is via a "HALT" instruction.

There are two methods of using software instructions to clear the Watchdog Timer, one of which must be chosen by configuration option. The first option is to use the single "CLR WDT" instruction while the second is to use the two commands "CLR WDT1" and "CLR WDT2". For the first option, a simple execution of "CLR WDT" will clear the WDT while for the second option, both "CLR WDT1" and "CLR WDT2" must both be executed to successfully clear the WDT. Note that for this second option, if "CLR WDT1" is used to clear the WDT, successive executions of this instruction will have no effect, only the execution of a "CLR WDT2" instruction will clear the WDT. Similarly, after the "CLR WDT2" instruction has been executed, only a successive "CLR WDT1" instruction can clear the Watchdog Timer.



Watchdog Timer Register



Watchdog Timer

Voice Output

Voice Control

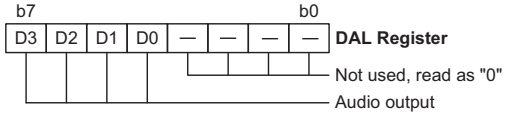
The voice control register controls the voice ROM circuit and the DAC circuit and selects the Voice ROM latch counter. If the DAC circuit is not enabled, any DAH/DAL outputs will be invalid. Writing a "1" to the DAC bit will enable the enable DAC circuit, while writing a "0" to the DAC bit will disable the DAC circuit. If the voice ROM circuit is not enabled, then voice ROM data cannot be accessed. Writing a "1" to the VROMC bit will enable the voice ROM circuit, while writing a "0" to the VROMC bit is will disable the voice ROM circuit. The LATCH bit determines which voice ROM address latch counter will be used as the voice ROM address latch counter.

Audio Output and Volume Control – DAL, DAH, VOL

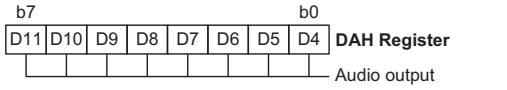
The audio output is 12-bits wide whose highest 8-bits are written into the DAH register and whose lowest four bits are written into the highest four bits of the DAL register. Bits 0~3 of the DAL register are always read as zero. There are 8 levels of volume which are setup using the VOL register. Only the highest 3-bits of this register are used for volume control, the other bits are not used and read as zero.

Voice ROM Data Address Latch Counter

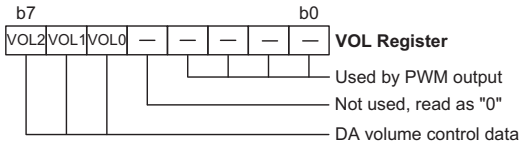
The Voice ROM address is 22-bits wide (except for the HT86B03 which has only 10-bits) and therefore requires three registers to store the address. There are two sets of three registers to store this address, which are LATCH0H/LATCH0M/LATCH0L and LATCH1H/LATCH1M/LATCH1L. The 22-bit address (except for the HT86B03 which has only 10-bits) stored in one set of these three registers is used to access the 8-bit voice code data in the Voice ROM. After the 8-bit Voice ROM data is addressed, a few instruction cycles, of at least 4us duration, are needed to latch the Voice ROM data. After this the microcontroller can read the voice data from the LATCHD register.



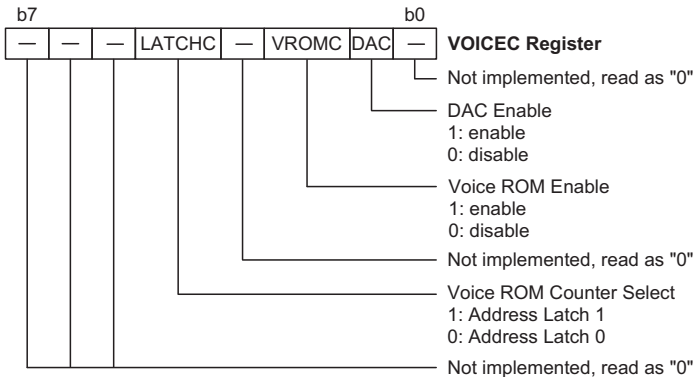
Digital to Analog Data Low Register



Digital to Analog Data High Register



Volume Control Register



VOICE Control Register

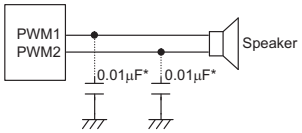
Example: Read an 8-bit voice ROM data which is located at address 000007H by address latch 0

```
Set       [26H].2       ; Enable voice ROM circuit
mov       A, 07H        ;
mov       LATCH0L, A    ; Set LATCH0L to 07H
mov       A, 00H        ;
mov       LATCH0M, A    ; Set LATCH0M to 00H
mov       A, 00H        ;
mov       LATCH0H, A    ; Set LATCH0H to 00H
call      Delay         ; Delay a short period of time
mov       A, LATCHD     ; Get voice data at 000007H
```

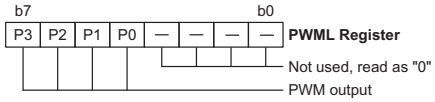
Pulse Width Modulation Output

All device include a single 12-bit PWM function. The PWM output is provided on two complimentary outputs on the PWM1 and PWM2 pins. These two pins can directly drive a piezo buzzer or an 8 ohm speaker without requiring any external components. The PWM1 output can also be used alone to drive a piezo buzzer or an 8 ohm speaker without requiring external components. When the single PWM1 output is chosen, which is achieved by setting the Single_PWM bit in the PWMC register.

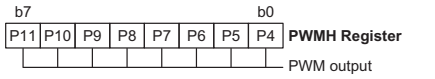
The PWM output will initially be at a low level, and if stopped will also return to a low level. If the PWMCC bit changes from low to high then the PWM function will start and latch new data. If the data is not updated then the old value will remain. If the PWMCC bit changes from high to low, at the end of the duty cycle, the PWM output will stop.



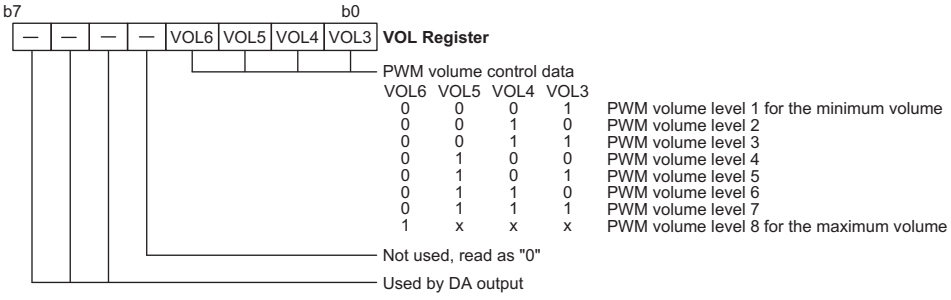
Note: "*" For reducing the digital noise that PWM may cause, can consider increment capacitors.



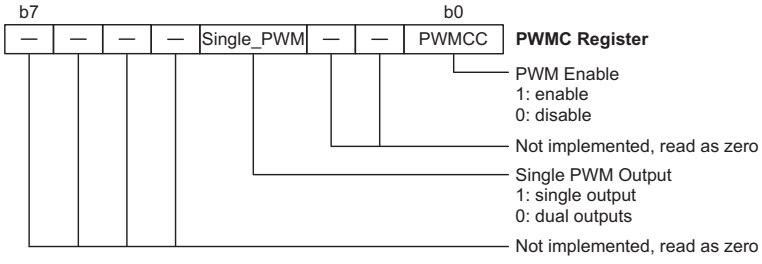
Pulse Width Modulator Data Low Register



Pulse Width Modulator Data High Register



Volume Control Register



Pulse Width Modulator Control Register

External RC Oscillation Converter

An external RC oscillation converter is implemented in certain devices and is a function which allows analog switch functions to be implemented. When used in conjunction with the Analog Switch function up to eight C/R-F can be implemented.

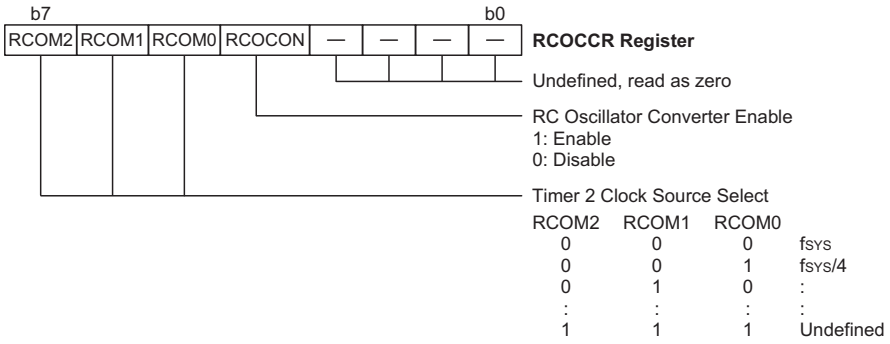
External RC Oscillation Converter Operation

The RC oscillation converter is composed of two 16-bit count-up programmable timers. One is Timer 2, described in the Timer section and the other is an additional counter known as Timer 4. The RC oscillation converter is enabled when the RCO bit, which is bit 1 of the RCOCR register, is set high. The RC oscillation converter will then be composed of four registers, TMR2L, TMR2H, TMR4L and TMR4H. The Timer 2 clock source comes from the system clock or from the system clock/4, the choice of which is determined by bits in the RCOCCR register. The RC oscillation converter Timer 4 clock source comes from an external RC oscillator. As the oscillation frequency is dependent upon external capacitance and resistance values, it can therefore be used to detect the increased capacitance of a analog switch pad.

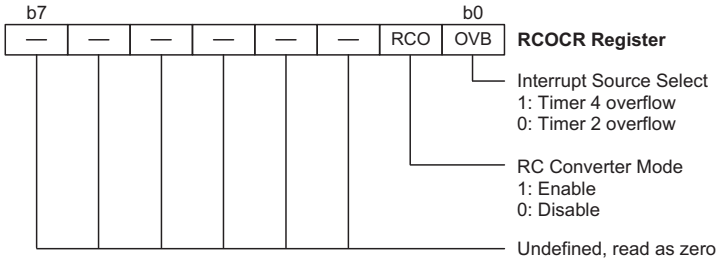
There are six registers related to the RC oscillation converter. These are, TMR2H, TMR2L, RCOCCR, TMR4H, TMR4L and RCOCR. The internal timer clock is the input clock source for TMR2H and TMR2L, while the external RC oscillator is the clock source input to TMR4H and TMR4L. The OVB bit, which is bit 0 of the RCOCR register, decides whether the timer interrupt is sourced from either the Timer 2 overflows or Timer 4 overflow. When a timer overflow occurs, the T2F bit is set and an external RC oscillation converter interrupt occurs. When the RC oscillation converter Timer 2 or Timer 4 overflows, the RCOCON bit is automatically reset to zero and stops counting.

The resistor and capacitor form an oscillation circuit and input to TMR4H and TMR4L. The RCOM0, RCOM1 and RCOM2 bits of RCOCCR define the clock source of Timer 2.

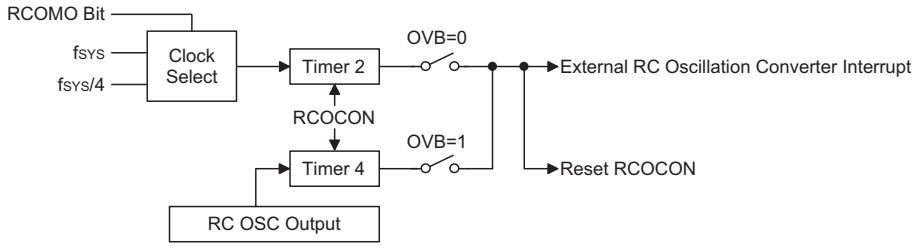
When the RCOCON bit, which is bit 4 of the RCOCCR register, is set high, Timer 2 and Timer 4 will start counting until Timer 2 or Timer 4 overflows. Now the timer counter will generate an interrupt request flag which is bit T2F, bit 4 of the INTCH register. Both Timer 2 and Timer 4 will then stop counting and the RCOCON bit will automatically be reset to "0" at the same time. Note that if the RCOCON bit is high, the TMR2H, TMR2L, TMR4H and TMR4L registers cannot be read or written to.



RCOCCR Register



RCOCR Register



Programming Considerations

As the 16-bit Timers have both low byte and high byte timer registers, accessing these registers is carried out in a specific way. It must be noted that when using instructions to preload data into the low byte registers, namely TMR2L or TMR4L, the data will only be placed into a low byte buffer and not directly into the low byte register. The actual transfer of the data into the low byte register is only carried out when a write to its associated high byte register, namely TMR2H or TMR4H, is executed. However, using instructions to preload data into the high byte timer register will result in the data being directly written to the high byte register. At the same time the data in the low byte buffer will be transferred into its associated low byte register. For this reason, when preloading data into the 16-bit timer registers, the low byte should be written first. It must also be noted that to read the contents of the low byte register, a read to the high byte register must first be executed to latch the contents of the low byte buffer into its associated low byte register. After this has been done, the low byte register can be read in the normal way. Note that reading the low byte timer register will only result in reading the previously latched contents of the low byte buffer and not the actual contents of the low byte timer register.

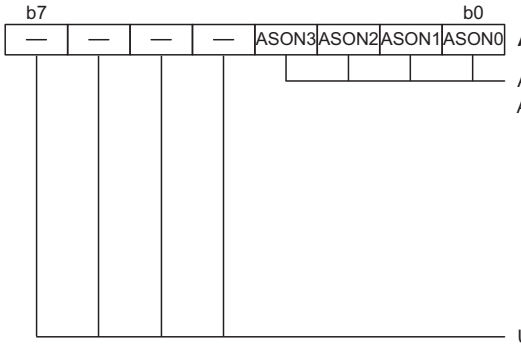
Program Example

External RC oscillation converter mode example program – Timer 2 overflow:

```
clr         RCOCCR
mov         a, 00000010b        ; Enable External RC oscillation mode and set Timer 2
                                ; overflow interrupt
mov         RCOCR,a
clr         intch.4             ; Clear External RC Oscillation Converter interrupt
                                ; request flag
mov         a, low (65536-1000); Give timer 2 initial value
mov         Tmr2l, a            ; Timer 2 count 1000 time and then overflow
mov         a, high (65536-1000)
mov         Tmr2h, a
mov         a, 00h              ; Give timer 4 initial value
mov         Tmr4l, a
mov         a, 00h
mov         Tmr4h, a
mov         a, 00110000b        ; Timer 2 clock source=fSYS/4 and timer on
mov         RCOCCR, a
p10:
clr         Wdt
snz         intch.4             ; Polling External RC Oscillation Converter interrupt
                                ; request flag
jmp         p10
clr         intch.4             ; Clear External RC Oscillation Converter interrupt
                                ; request flag
                                ; Program continue
```

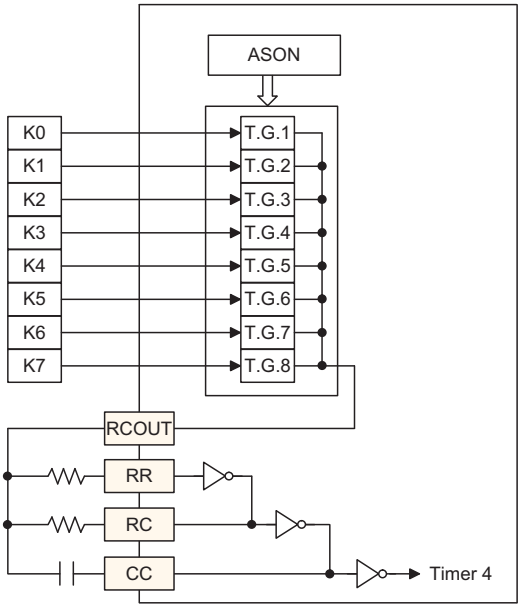
Analog Switch

There are 8 analog switch lines in the microcontroller, labeled as K0 ~ K7, and the Analog Switch control register, which is mapped to the data memory by option. All of these Analog Switch lines can be used together with the external RC Oscillation Converter for C/R-F input keys.



ASON3	ASON2	ASON1	ASON0	
0	0	0	0	K0 on, others off
0	0	0	1	K1 on, others off
0	0	1	0	K2 on, others off
0	0	1	1	K3 on, others off
0	1	0	0	K4 on, others off
0	1	0	1	K5 on, others off
0	1	1	0	K6 on, others off
0	1	1	1	K7 on, others off
1	X	X	X	All off, OSC off

Analog Switch Control Register – ASCR



Analog Switch

Configuration Options

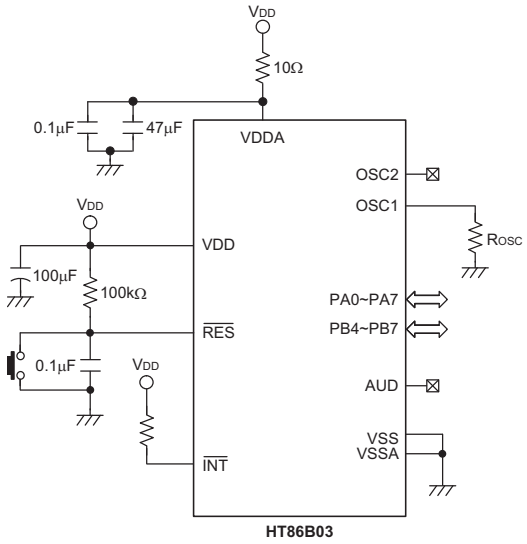
Configuration options refer to certain options within the MCU that are programmed into the device during the programming process. During the development process, these options are selected using the HT-IDE software development tools. As these options are programmed into the device using the hardware programming tools, once they are selected they cannot be changed later by the application software.

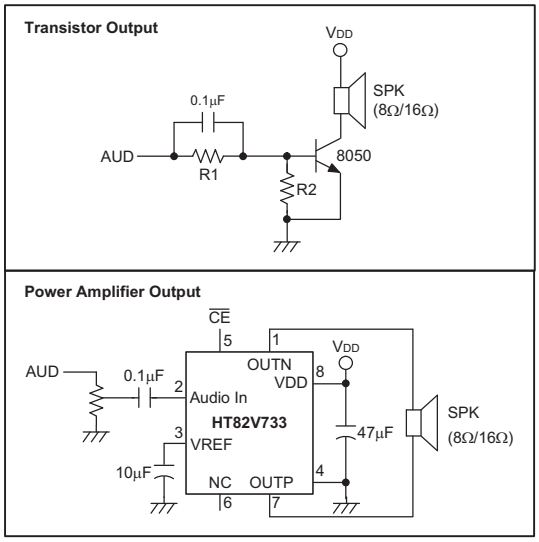
No.	HT86B03/HT86B10/HT86BR10/HT86B20/HT86B30/HT86BR30 Options
I/O Options	
1	PA0~PA7: wake-up enable or disable (bit option)
2	PA0~PA7: pull-high enable or disable (bit option)
3	PB0~PB7: pull-high enable or disable (bit option) - the HT86B03 device only has PB4~PB7
Oscillation Option	
4	OSC type selection: RC or crystal
Interrupt Option	
5	INT Triggering edge: Falling or both
Watchdog Options	
6	WDT: enable or disable
7	WDT clock source: WDROSC or T1
8	CLRWDT instructions: 1 or 2 instructions
Low Voltage Reset Option	
9	LVR select: enable or disable

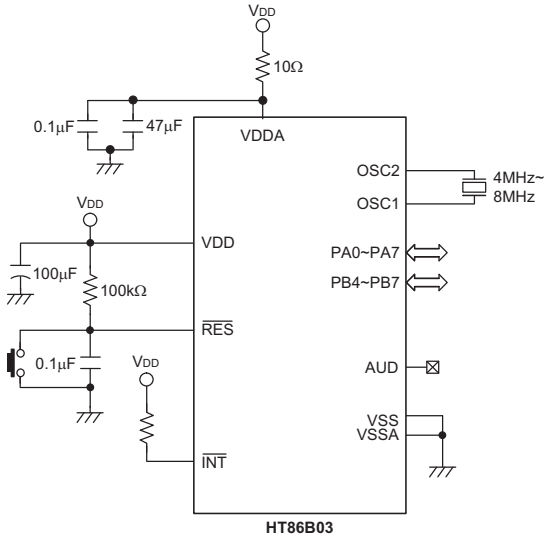
No.	HT86B40/HT86B50/HT86B60/HT86BR60/HT86B70/HT86B80/HT86B90 Options
I/O Options	
1	PA0~PA7: wake-up enable or disable
2	PA0~PA7: pull-high enable or disable
3	PB0~PB7: pull-high enable or disable
4	PD0~PD7: pull-high enable or disable
5	PB share pin select: PB0~7 or K0~7
6	PD share pin select: PD4~7 or external RC oscillation converter pin
Oscillation Option	
7	OSC type selection: RC or crystal
Interrupt Option	
8	INT Triggering edge: Falling or both
Watchdog Options	
9	WDT: enable or disable
10	WDT clock source: WDROSC or T1
11	CLRWDT instructions: 1 or 2 instructions
Low Voltage Reset Option	
12	LVR select: enable or disable

Application Circuits

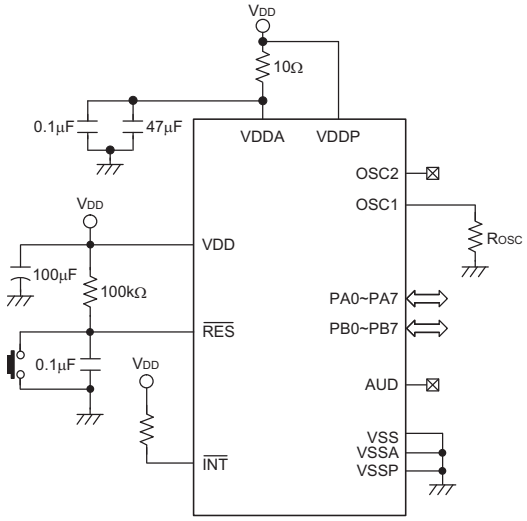
HT86B03



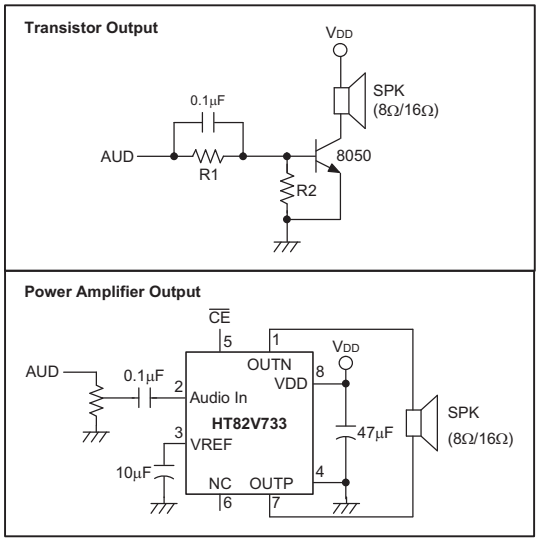


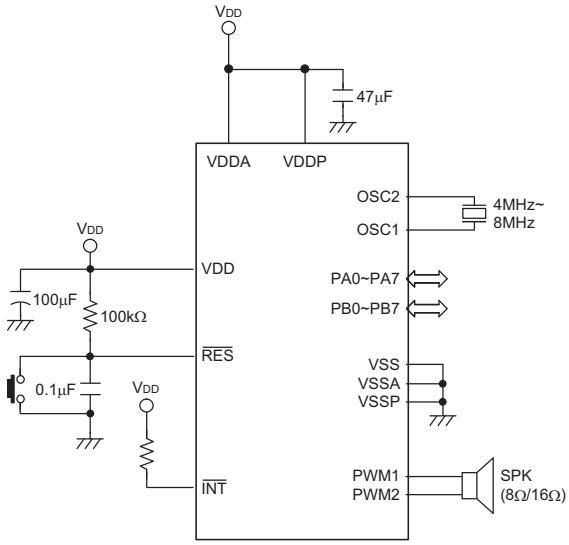


HT86B10/HT86BR10/HT86B20/HT86B30/HT86BR30



HT86B10/HT86BR10
HT86B20/HT86B30/HT86BR30

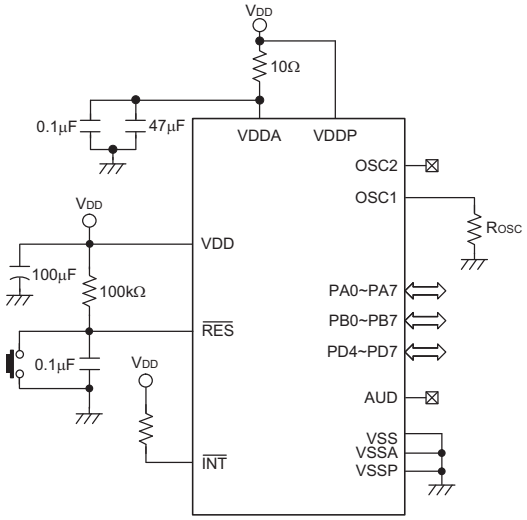




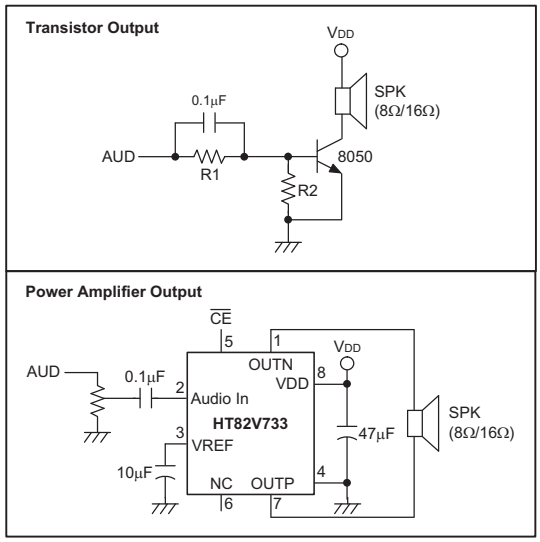
HT86B10/HT86BR10
HT86B20/HT86B30/HT86BR30

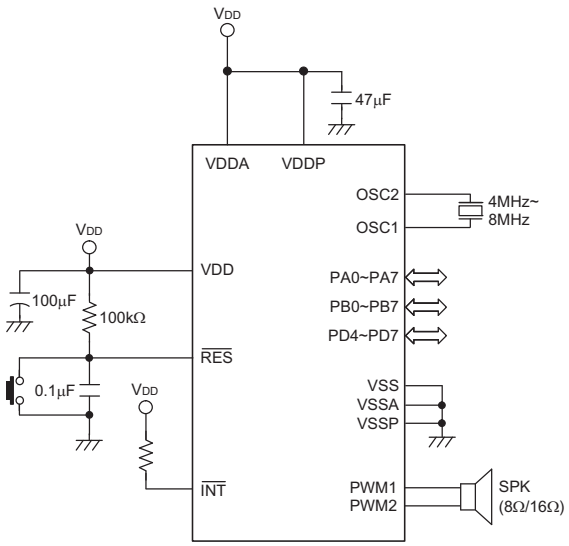
Note: The PWM application refer to the description of Pulse Width Modulation Output.

HT86B40/HT86B50/HT86B60/HT86BR60



HT86B40/HT86B50/HT86B60/HT86BR60

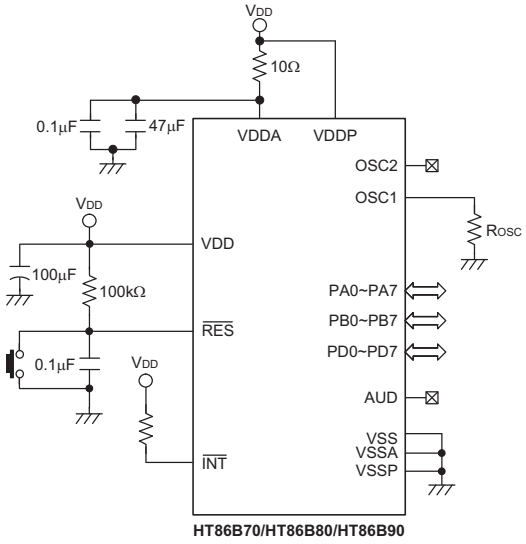




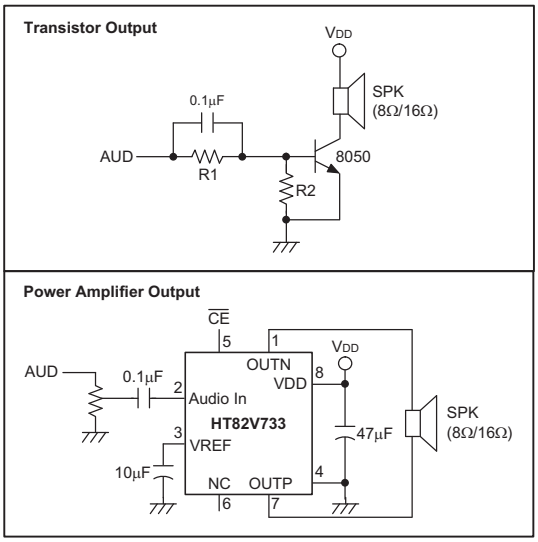
HT86B40/HT86B50/HT86B60/HT86BR60

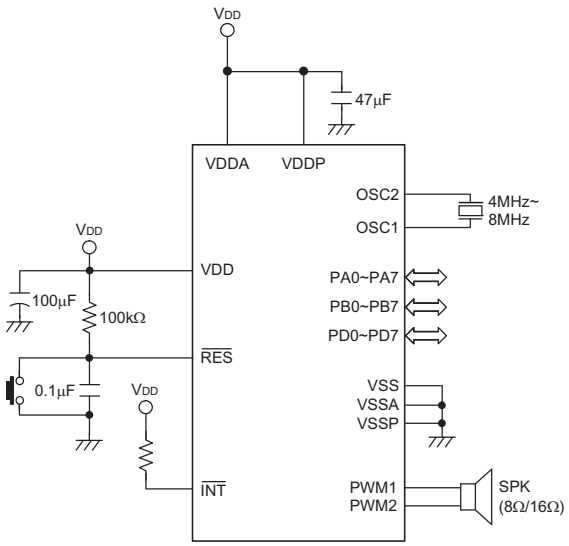
Note: The PWM application refer to the description of Pulse Width Modulation Output.

HT86B70/HT86B80/HT86B90



HT86B70/HT86B80/HT86B90





HT86B70/HT86B80/HT86B90

Note: The PWM application refer to the description of Pulse Width Modulation Output.

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 microcontrollers, 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 Operations

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 where 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 setup as a table where data can be directly stored. A set of easy to use instructions provides the means by which this fixed data can be referenced and retrieved from the Program Memory.

Other Operations

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

Instruction Set Summary

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

Table conventions:

x: Bits immediate data

m: Data Memory address

A: Accumulator

i: 0~7 number of bits

addr: Program memory address

Mnemonic	Description	Cycles	Flag Affected
Arithmetic			
ADD A,[m]	Add Data Memory to ACC	1	Z, C, AC, OV
ADDM A,[m]	Add ACC to Data Memory	1Note	Z, C, AC, OV
ADD A,x	Add immediate data to ACC	1	Z, C, AC, OV
ADC A,[m]	Add Data Memory to ACC with Carry	1	Z, C, AC, OV
ADCM A,[m]	Add ACC to Data memory with Carry	1Note	Z, C, AC, OV
SUB A,x	Subtract immediate data from the ACC	1	Z, C, AC, OV
SUB A,[m]	Subtract Data Memory from ACC	1	Z, C, AC, OV
SUBM A,[m]	Subtract Data Memory from ACC with result in Data Memory	1Note	Z, C, AC, OV
SBC A,[m]	Subtract Data Memory from ACC with Carry	1	Z, C, AC, OV
SBCM A,[m]	Subtract Data Memory from ACC with Carry, result in Data Memory	1Note	Z, C, AC, OV
DAA [m]	Decimal adjust ACC for Addition with result in Data Memory	1Note	C
Logic Operation			
AND A,[m]	Logical AND Data Memory to ACC	1	Z
OR A,[m]	Logical OR Data Memory to ACC	1	Z
XOR A,[m]	Logical XOR Data Memory to ACC	1	Z
ANDM A,[m]	Logical AND ACC to Data Memory	1Note	Z
ORM A,[m]	Logical OR ACC to Data Memory	1Note	Z
XORM A,[m]	Logical XOR ACC to Data Memory	1Note	Z
AND A,x	Logical AND immediate Data to ACC	1	Z
OR A,x	Logical OR immediate Data to ACC	1	Z
XOR A,x	Logical XOR immediate Data to ACC	1	Z
CPL [m]	Complement Data Memory	1Note	Z
CPLA [m]	Complement Data Memory with result in ACC	1	Z
Increment & Decrement			
INCA [m]	Increment Data Memory with result in ACC	1	Z
INC [m]	Increment Data Memory	1Note	Z
DECA [m]	Decrement Data Memory with result in ACC	1	Z
DEC [m]	Decrement Data Memory	1Note	Z

Mnemonic	Description	Cycles	Flag Affected
Rotate			
RRA [m]	Rotate Data Memory right with result in ACC	1	None
RR [m]	Rotate Data Memory right	1Note	None
RRCA [m]	Rotate Data Memory right through Carry with result in ACC	1	C
RRC [m]	Rotate Data Memory right through Carry	1Note	C
RLA [m]	Rotate Data Memory left with result in ACC	1	None
RL [m]	Rotate Data Memory left	1Note	None
RLCA [m]	Rotate Data Memory left through Carry with result in ACC	1	C
RLC [m]	Rotate Data Memory left through Carry	1Note	C
Data Move			
MOV A,[m]	Move Data Memory to ACC	1	None
MOV [m],A	Move ACC to Data Memory	1Note	None
MOV A,x	Move immediate data to ACC	1	None
Bit Operation			
CLR [m].i	Clear bit of Data Memory	1Note	None
SET [m].i	Set bit of Data Memory	1Note	None
Branch			
JMP addr	Jump unconditionally	2	None
SZ [m]	Skip if Data Memory is zero	1Note	None
SZA [m]	Skip if Data Memory is zero with data movement to ACC	1note	None
SZ [m].i	Skip if bit i of Data Memory is zero	1Note	None
SNZ [m].i	Skip if bit i of Data Memory is not zero	1Note	None
SIZ [m]	Skip if increment Data Memory is zero	1Note	None
SDZ [m]	Skip if decrement Data Memory is zero	1Note	None
SIZA [m]	Skip if increment Data Memory is zero with result in ACC	1Note	None
SDZA [m]	Skip if decrement Data Memory is zero with result in ACC	1Note	None
CALL addr	Subroutine call	2	None
RET	Return from subroutine	2	None
RET A,x	Return from subroutine and load immediate data to ACC	2	None
RETI	Return from interrupt	2	None
Table Read			
TABRDC [m]	Read table (current page) to TBLH and Data Memory	2Note	None
TABRDL [m]	Read table (last page) to TBLH and Data Memory	2Note	None
Miscellaneous			
NOP	No operation	1	None
CLR [m]	Clear Data Memory	1Note	None
SET [m]	Set Data Memory	1Note	None
CLR WDT	Clear Watchdog Timer	1	TO, PDF
CLR WDT1	Pre-clear Watchdog Timer	1	TO, PDF
CLR WDT2	Pre-clear Watchdog Timer	1	TO, PDF
SWAP [m]	Swap nibbles of Data Memory	1Note	None
SWAPA [m]	Swap nibbles of Data Memory with result in ACC	1	None
HALT	Enter power down mode	1	TO, PDF

Note: 1. For skip instructions, if the result of the comparison involves a skip then two cycles are required, if no skip takes place only one cycle is required.
2. Any instruction which changes the contents of the PCL will also require 2 cycles for execution.
3. For the ″CLR WDT1″ and ″CLR WDT2″ instructions the TO and PDF flags may be affected by the execution status. The TO and PDF flags are cleared after both ″CLR WDT1″ and ″CLR WDT2″ instructions are consecutively executed. Otherwise the TO and PDF flags remain unchanged.

Instruction Definition

ADC A,[m]	Add Data Memory to ACC with Carry
Description	The contents of the specified Data Memory, Accumulator and the carry flag are added. The result is stored in the Accumulator.
Operation	ACC ← ACC + [m] + C
Affected flag(s)	OV, Z, AC, C
ADCM A,[m]	Add ACC to Data Memory with Carry
Description	The contents of the specified Data Memory, Accumulator and the carry flag are added. The result is stored in the specified Data Memory.
Operation	[m] ← ACC + [m] + C
Affected flag(s)	OV, Z, AC, C
ADD A,[m]	Add Data Memory to ACC
Description	The contents of the specified Data Memory and the Accumulator are added. The result is stored in the Accumulator.
Operation	ACC ← ACC + [m]
Affected flag(s)	OV, Z, AC, C
ADD A,x	Add immediate data to ACC
Description	The contents of the Accumulator and the specified immediate data are added. The result is stored in the Accumulator.
Operation	ACC ← ACC + x
Affected flag(s)	OV, Z, AC, C
ADDM A,[m]	Add ACC to Data Memory
Description	The contents of the specified Data Memory and the Accumulator are added. The result is stored in the specified Data Memory.
Operation	[m] ← ACC + [m]
Affected flag(s)	OV, Z, AC, C
AND A,[m]	Logical AND Data Memory to ACC
Description	Data in the Accumulator and the specified Data Memory perform a bitwise logical AND operation. The result is stored in the Accumulator.
Operation	ACC ← ACC ″AND″ [m]
Affected flag(s)	Z
AND A,x	Logical AND immediate data to ACC
Description	Data in the Accumulator and the specified immediate data perform a bitwise logical AND operation. The result is stored in the Accumulator.
Operation	ACC ← ACC ″AND″ x
Affected flag(s)	Z
ANDM A,[m]	Logical AND ACC to Data Memory
Description	Data in the specified Data Memory and the Accumulator perform a bitwise logical AND operation. The result is stored in the Data Memory.
Operation	[m] ← ACC ″AND″ [m]
Affected flag(s)	Z

CALL addr	Subroutine call
Description	Unconditionally calls a subroutine at the specified address. The Program Counter then increments by 1 to obtain the address of the next instruction which is then pushed onto the stack. The specified address is then loaded and the program continues execution from this new address. As this instruction requires an additional operation, it is a two cycle instruction.
Operation	Stack ← Program Counter + 1 Program Counter ← addr
Affected flag(s)	None
CLR [m]	Clear Data Memory
Description	Each bit of the specified Data Memory is cleared to 0.
Operation	[m] ← 00H
Affected flag(s)	None
CLR [m].i	Clear bit of Data Memory
Description	Bit i of the specified Data Memory is cleared to 0.
Operation	[m].i ← 0
Affected flag(s)	None
CLR WDT	Clear Watchdog Timer
Description	The TO, PDF flags and the WDT are all cleared.
Operation	WDT cleared TO ← 0 PDF ← 0
Affected flag(s)	TO, PDF
CLR WDT1	Pre-clear Watchdog Timer
Description	The TO, PDF flags and the WDT are all cleared. Note that this instruction works in conjunction with CLR WDT2 and must be executed alternately with CLR WDT2 to have effect. Repetitively executing this instruction without alternately executing CLR WDT2 will have no effect.
Operation	WDT cleared TO ← 0 PDF ← 0
Affected flag(s)	TO, PDF
CLR WDT2	Pre-clear Watchdog Timer
Description	The TO, PDF flags and the WDT are all cleared. Note that this instruction works in conjunction with CLR WDT1 and must be executed alternately with CLR WDT1 to have effect. Repetitively executing this instruction without alternately executing CLR WDT1 will have no effect.
Operation	WDT cleared TO ← 0 PDF ← 0
Affected flag(s)	TO, PDF

CPL [m] Complement Data Memory

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

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

Affected flag(s) Z

CPLA [m] Complement Data Memory with result in ACC

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

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

Affected flag(s) Z

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

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

Operation $[m] \leftarrow ACC + 00H$ or
$[m] \leftarrow ACC + 06H$ or
$[m] \leftarrow ACC + 60H$ or
$[m] \leftarrow ACC + 66H$

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 $TO \leftarrow 0$
$PDF \leftarrow 1$

Affected flag(s) TO, PDF

INC [m]	Increment Data Memory
Description	Data in the specified Data Memory is incremented by 1.
Operation	[m] ← [m] + 1
Affected flag(s)	Z
INCA [m]	Increment Data Memory with result in ACC
Description	Data in the specified Data Memory is incremented by 1. The result is stored in the Accumulator. The contents of the Data Memory remain unchanged.
Operation	ACC ← [m] + 1
Affected flag(s)	Z
JMP addr	Jump unconditionally
Description	The contents of the Program Counter are replaced with the specified address. Program execution then continues from this new address. As this requires the insertion of a dummy instruction while the new address is loaded, it is a two cycle instruction.
Operation	Program Counter ← addr
Affected flag(s)	None
MOV A,[m]	Move Data Memory to ACC
Description	The contents of the specified Data Memory are copied to the Accumulator.
Operation	ACC ← [m]
Affected flag(s)	None
MOV A,x	Move immediate data to ACC
Description	The immediate data specified is loaded into the Accumulator.
Operation	ACC ← x
Affected flag(s)	None
MOV [m],A	Move ACC to Data Memory
Description	The contents of the Accumulator are copied to the specified Data Memory.
Operation	[m] ← ACC
Affected flag(s)	None
NOP	No operation
Description	No operation is performed. Execution continues with the next instruction.
Operation	No operation
Affected flag(s)	None
OR A,[m]	Logical OR Data Memory to ACC
Description	Data in the Accumulator and the specified Data Memory perform a bitwise logical OR operation. The result is stored in the Accumulator.
Operation	ACC ← ACC ″OR″ [m]
Affected flag(s)	Z

OR A,x	Logical OR immediate data to ACC
Description	Data in the Accumulator and the specified immediate data perform a bitwise logical OR operation. The result is stored in the Accumulator.
Operation	ACC ← ACC ″OR″ x
Affected flag(s)	Z
ORM A,[m]	Logical OR ACC to Data Memory
Description	Data in the specified Data Memory and the Accumulator perform a bitwise logical OR operation. The result is stored in the Data Memory.
Operation	[m] ← ACC ″OR″ [m]
Affected flag(s)	Z
RET	Return from subroutine
Description	The Program Counter is restored from the stack. Program execution continues at the restored address.
Operation	Program Counter ← Stack
Affected flag(s)	None
RET A,x	Return from subroutine and load immediate data to ACC
Description	The Program Counter is restored from the stack and the Accumulator loaded with the specified immediate data. Program execution continues at the restored address.
Operation	Program Counter ← Stack ACC ← x
Affected flag(s)	None
RETI	Return from interrupt
Description	The Program Counter is restored from the stack and the interrupts are re-enabled by setting the EMI bit. EMI is the master interrupt global enable bit. If an interrupt was pending when the RETI instruction is executed, the pending Interrupt routine will be processed before returning to the main program.
Operation	Program Counter ← Stack EMI ← 1
Affected flag(s)	None
RL [m]	Rotate Data Memory left
Description	The contents of the specified Data Memory are rotated left by 1 bit with bit 7 rotated into bit 0.
Operation	[m].(i+1) ← [m].i; (i = 0~6) [m].0 ← [m].7
Affected flag(s)	None
RLA [m]	Rotate Data Memory left with result in ACC
Description	The contents of the specified Data Memory are rotated left by 1 bit with bit 7 rotated into bit 0. The rotated result is stored in the Accumulator and the contents of the Data Memory remain unchanged.
Operation	ACC.(i+1) ← [m].i; (i = 0~6) ACC.0 ← [m].7
Affected flag(s)	None

RLC [m] Rotate Data Memory left through Carry

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

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

Affected flag(s) C

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

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

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

Affected flag(s) C

RR [m] Rotate Data Memory right

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

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

Affected flag(s) None

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

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

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

Affected flag(s) None

RRC [m] Rotate Data Memory right through Carry

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

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

Affected flag(s) C

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

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

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

Affected flag(s) C

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

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

Operation $ACC \leftarrow ACC - [m] - \overline{C}$

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 $[m] \leftarrow [m] - 1$
Skip if $[m] = 0$

Affected flag(s) None

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

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

Operation $ACC \leftarrow [m] - 1$
Skip if $ACC = 0$

Affected flag(s) None

SET [m] Set Data Memory

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

Operation $[m] \leftarrow FFH$

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 $[m].i \leftarrow 1$

Affected flag(s) None

SIZ [m] Skip if increment Data Memory is 0

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

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

Affected flag(s) None

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

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

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

Affected flag(s) None

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

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

Operation Skip if [m].i $\neq$ 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 ← ACC – [m]

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

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

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

Operation [m] ← ACC – [m]

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

SUB A,x Subtract immediate data from ACC

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

Operation ACC ← ACC – x

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

SWAP [m] Swap nibbles of Data Memory

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

Operation [m].3~[m].0 ↔ [m].7 ~ [m].4

Affected flag(s) None

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

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

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

Affected flag(s) None

SZ [m] Skip if Data Memory is 0

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

Operation Skip if [m] = 0

Affected flag(s) None

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

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

Operation ACC ← [m]
Skip if [m] = 0

Affected flag(s) None

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

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

Operation Skip if [m].i = 0

Affected flag(s) None

TABRDC [m] Read table (current page) to TBLH and Data Memory

Description The low byte of the program code (current 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

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

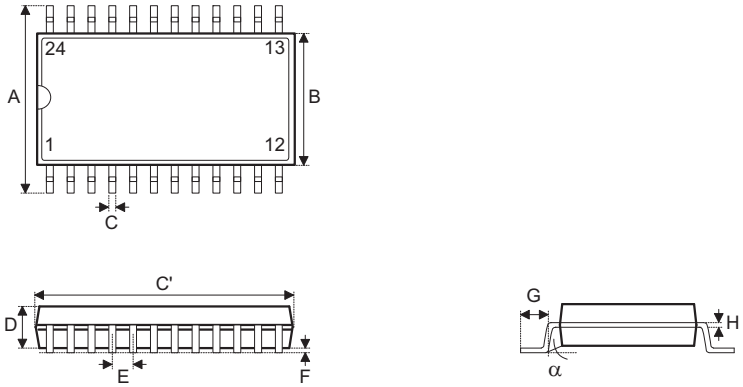
XOR A,[m]	Logical XOR Data Memory to ACC
Description	Data in the Accumulator and the specified Data Memory perform a bitwise logical XOR operation. The result is stored in the Accumulator.
Operation	ACC ← ACC ″XOR″ [m]
Affected flag(s)	Z

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

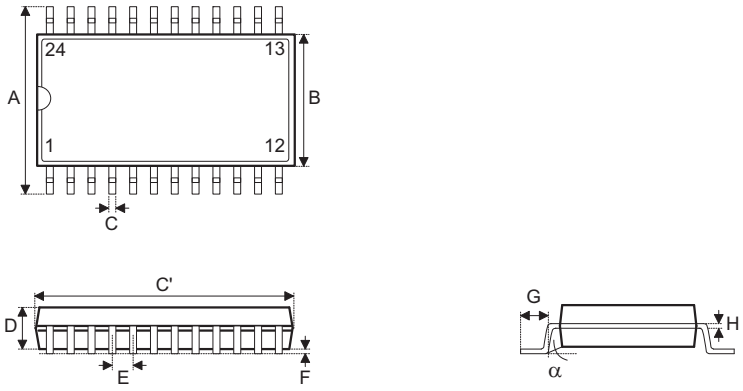
24-pin SSOP (150mil) Outline Dimensions



Symbol	Dimensions in inch		
	Min.	Nom.	Max.
A	0.228	—	0.244
B	0.150	—	0.157
C	0.008	—	0.012
C′	0.335	—	0.346
D	0.054	—	0.060
E	—	0.025	—
F	0.004	—	0.010
G	0.022	—	0.028
H	0.007	—	0.010
α	0°	—	8°

Symbol	Dimensions in mm		
	Min.	Nom.	Max.
A	5.79	—	6.20
B	3.81	—	3.99
C	0.20	—	0.30
C′	8.51	—	8.79
D	1.37	—	1.52
E	—	0.64	—
F	0.10	—	0.25
G	0.56	—	0.71
H	0.18	—	0.25
α	0°	—	8°

24-pin SSOP (209mil) Outline Dimensions

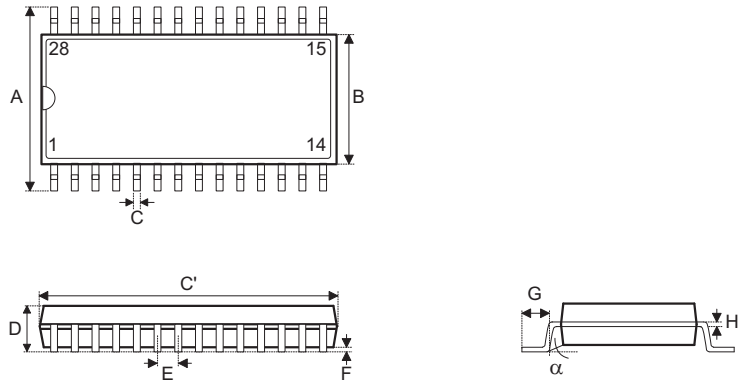


• MO-150

Symbol	Dimensions in inch		
	Min.	Nom.	Max.
A	0.291	—	0.323
B	0.197	—	0.220
C	0.009	—	0.013
C′	0.311	—	0.335
D	—	—	0.079
E	—	0.026	—
F	0.002	—	—
G	0.022	—	0.037
H	0.004	—	0.008
α	0°	—	8°

Symbol	Dimensions in mm		
	Min.	Nom.	Max.
A	7.40	—	8.20
B	5.00	—	5.60
C	0.22	—	0.33
C′	7.90	—	8.50
D	—		2.00
E	—	0.65	—
F	0.05	—	—
G	0.55	—	0.95
H	0.09	—	0.21
α	0°	—	8°

28-pin SOP (300mil) Outline Dimensions

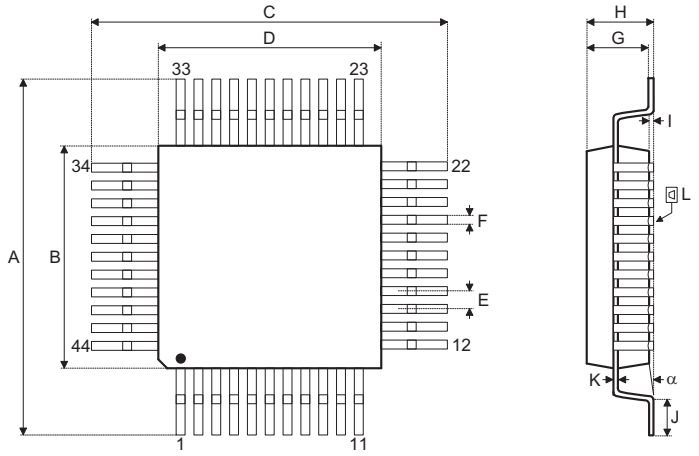


• MS-013

Symbol	Dimensions in inch		
	Min.	Nom.	Max.
A	0.393	—	0.419
B	0.256	—	0.300
C	0.012	—	0.020
C′	0.697	—	0.713
D	—	—	0.104
E	—	0.050	—
F	0.004	—	0.012
G	0.016	—	0.050
H	0.008	—	0.013
α	0°	—	8°

Symbol	Dimensions in mm		
	Min.	Nom.	Max.
A	9.98	—	10.64
B	6.50	—	7.62
C	0.30	—	0.51
C′	17.70	—	18.11
D	—	—	2.64
E	—	1.27	—
F	0.10	—	0.30
G	0.41	—	1.27
H	0.20	—	0.33
α	0°	—	8°

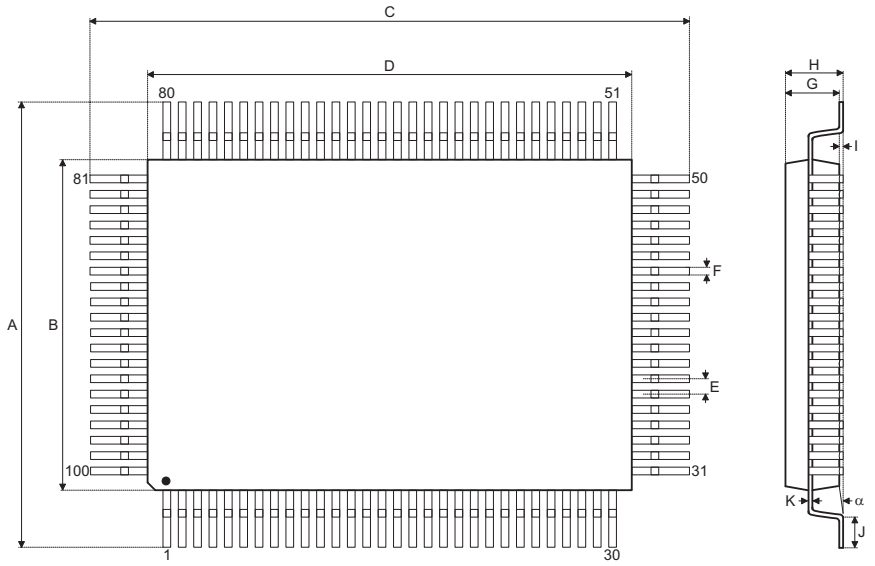
44-pin QFP (10mm×10mm) Outline Dimensions



Symbol	Dimensions in inch		
	Min.	Nom.	Max.
A	0.512	—	0.528
B	0.390	—	0.398
C	0.512	—	0.528
D	0.390	—	0.398
E	—	0.031	—
F	—	0.012	—
G	0.075	—	0.087
H	—	—	0.106
I	0.010	—	0.020
J	0.029	—	0.037
K	0.004	—	0.008
L	—	0.004	—
α	0°	—	7°

Symbol	Dimensions in mm		
	Min.	Nom.	Max.
A	13.00	—	13.40
B	9.90	—	10.10
C	13.00	—	13.40
D	9.90	—	10.10
E	—	0.80	—
F	—	0.30	—
G	1.90	—	2.20
H	—	—	2.70
I	0.25	—	0.50
J	0.73	—	0.93
K	0.10	—	0.20
L	—	0.10	—
α	0°	—	7°

100-pin QFP (14mm×20mm) Outline Dimensions

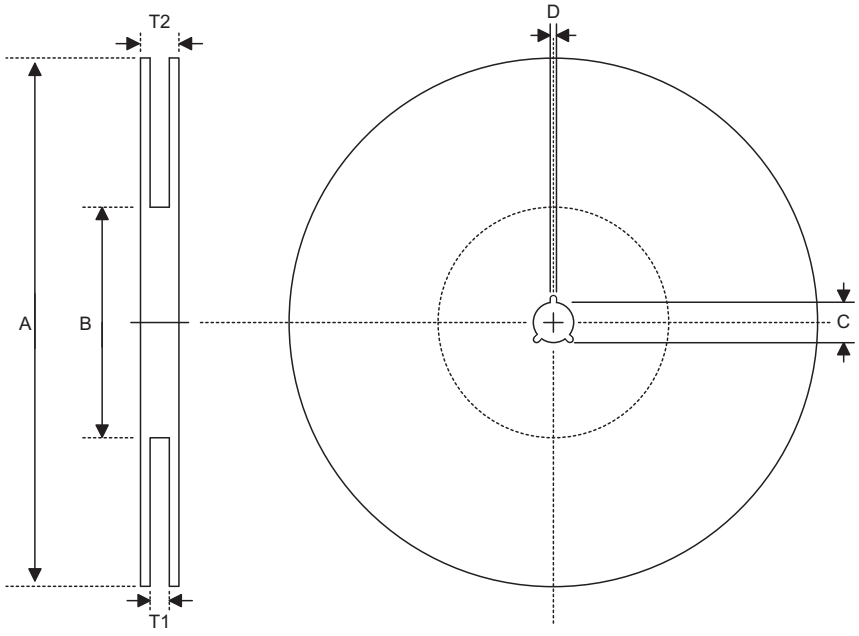


Symbol	Dimensions in inch		
	Min.	Nom.	Max.
A	0.728	—	0.756
B	0.547	—	0.555
C	0.965	—	0.992
D	0.783	—	0.791
E	—	0.026	—
F	—	0.012	—
G	0.098	—	0.122
H	—	—	0.134
I	—	0.004	—
J	0.039	—	0.055
K	0.004	—	0.008
α	0°	—	7°

Symbol	Dimensions in mm		
	Min.	Nom.	Max.
A	18.50	—	19.20
B	13.90	—	14.10
C	24.50	—	25.20
D	19.90	—	20.10
E	—	0.65	—
F	—	0.30	—
G	2.50	—	3.10
H	—	—	3.40
I	—	0.1	—
J	1.00	—	1.40
K	0.10	—	0.20
α	0°	—	7°

Product Tape and Reel Specifications

Reel Dimensions



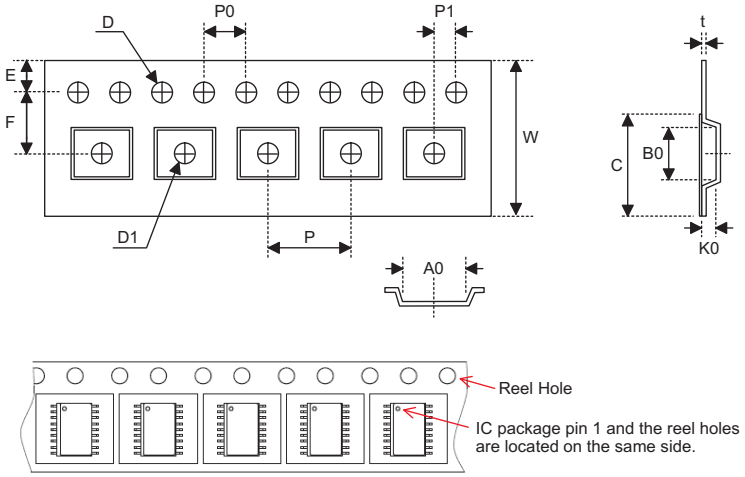
SSOP 24S (150mil)

Symbol	Description	Dimensions in mm
A	Reel Outer Diameter	330.0±1.0
B	Reel Inner Diameter	100.0±1.5
C	Spindle Hole Diameter	13.0 $^{+0.5/-0.2}$
D	Key Slit Width	2.0±0.5
T1	Space Between Flange	16.8 $^{+0.3/-0.2}$
T2	Reel Thickness	22.2±0.2

SOP 28W (300mil)

Symbol	Description	Dimensions in mm
A	Reel Outer Diameter	330.0±1.0
B	Reel Inner Diameter	100.0±1.5
C	Spindle Hole Diameter	13.0 $^{+0.5/-0.2}$
D	Key Slit Width	2.0±0.5
T1	Space Between Flange	24.8 $^{+0.3/-0.2}$
T2	Reel Thickness	30.2±0.2

Carrier Tape Dimensions



SSOP 24S (150mil)

Symbol	Description	Dimensions in mm
W	Carrier Tape Width	16.0 +0.3/-0.1
P	Cavity Pitch	8.0±0.1
E	Perforation Position	1.75±0.10
F	Cavity to Perforation (Width Direction)	7.5±0.1
D	Perforation Diameter	1.5 +0.1/-0.0
D1	Cavity Hole Diameter	1.5 +0.25/-0.0
P0	Perforation Pitch	4.0±0.1
P1	Cavity to Perforation (Length Direction)	2.0±0.1
A0	Cavity Length	6.5±0.1
B0	Cavity Width	9.5±0.1
K0	Cavity Depth	2.1±0.1
t	Carrier Tape Thickness	0.30±0.05
C	Cover Tape Width	13.3±0.1

SOP 28W (300mil)

Symbol	Description	Dimensions in mm
W	Carrier Tape Width	24.0±0.3
P	Cavity Pitch	12.0±0.1
E	Perforation Position	1.75±0.10
F	Cavity to Perforation (Width Direction)	11.5±0.1
D	Perforation Diameter	1.5 +0.1/-0.0
D1	Cavity Hole Diameter	1.5 +0.25/-0.0
P0	Perforation Pitch	4.0±0.1
P1	Cavity to Perforation (Length Direction)	2.0±0.1
A0	Cavity Length	10.85±0.10
B0	Cavity Width	18.34±0.10
K0	Cavity Depth	2.97±0.10
t	Carrier Tape Thickness	0.35±0.01
C	Cover Tape Width	21.3±0.1

Holtek Semiconductor Inc. (Headquarters)
No.3, Creation Rd. II, Science Park, Hsinchu, Taiwan
Tel: 886-3-563-1999
Fax: 886-3-563-1189
http://www.holtek.com.tw

Holtek Semiconductor Inc. (Taipei Sales Office)
4F-2, No. 3-2, YuanQu St., Nankang Software Park, Taipei 115, Taiwan
Tel: 886-2-2655-7070
Fax: 886-2-2655-7373
Fax: 886-2-2655-7383 (International sales hotline)

Holtek Semiconductor Inc. (Shenzhen Sales Office)
5F, Unit A, Productivity Building, No.5 Gaoxin M 2nd Road, Nanshan District, Shenzhen, China 518057
Tel: 86-755-8616-9908, 86-755-8616-9308
Fax: 86-755-8616-9722

Holtek Semiconductor (USA), Inc. (North America Sales Office)
46729 Fremont Blvd., Fremont, CA 94538
Tel: 1-510-252-9880
Fax: 1-510-252-9885
http://www.holtek.com

Copyright © 2010 by HOLTEK SEMICONDUCTOR INC.

The information appearing in this Data Sheet is believed to be accurate at the time of publication. However, Holtek assumes no responsibility arising from the use of the specifications described. The applications mentioned herein are used solely for the purpose of illustration and Holtek makes no warranty or representation that such applications will be suitable without further modification, nor recommends the use of its products for application that may present a risk to human life due to malfunction or otherwise. Holtek's products are not authorized for use as critical components in life support devices or systems. Holtek reserves the right to alter its products without prior notification. For the most up-to-date information, please visit our web site at http://www.holtek.com.tw.