HT3630
2 Channels Ad-lib Micro®

Features

- Operating voltage: 3.3V~5.2V
- System frequency: 3.58MHz
- An RC or crystal type of oscillator
- Low standby current
- 8 bit D/A audio output
- 8 bit μC
  - 2K × 14 bits of program memory (ROM)
  - 96 × 8 bits of data memory (RAM)
  - 2 stacks available
  - A pair of memory pointers
  - A watch dog timer
  - Interrupt input
  - An 8 stage timer
  - An 8 bit input/output port
  - 12 input lines
  - 64 powerful instructions
  - Halt function
  - 14 bit table read instruction
  - 2 level subroutine nesting
  - Bit manipulation instruction
- 2 channels of melody processing
  - 8 built-in timbre waveforms ($8 \times 64 \times 7$ bits, PCM code)
  - Envelope programmable
  - 5 octaves
- 4 channels of percussion
- 4 modes of main volume controllable

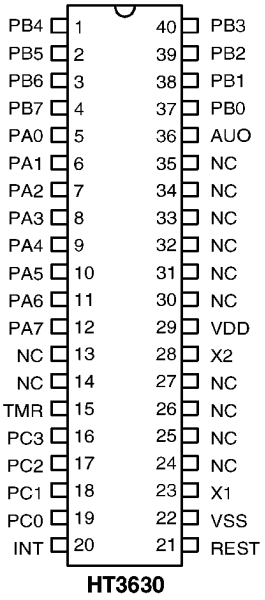
General Description

The HT3630 is a CMOS VLSI designed for musical processing. It is built with an 8 bit microcontroller, 2 channels ETS (Electronic Tone Synthesizer) in addition to a percussion generator circuit.

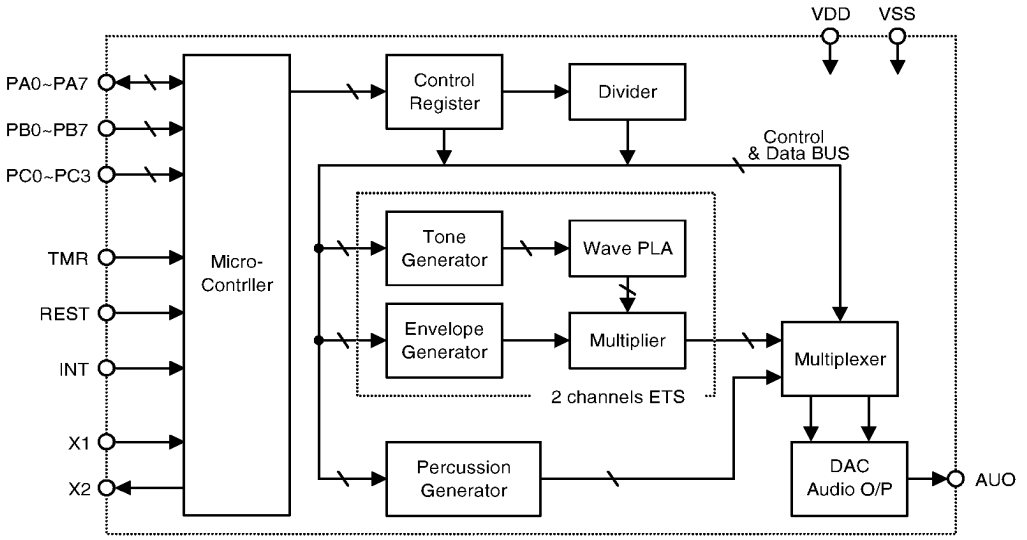
The μC of the VLSI contains a program ROM of 2K × 14 bits. The program ROM can be programmed to scan a keyboard of an instrument. The μC will start operation after receiving a key input, and then send out a control code to control the ETS or sound processing circuit. The ETS or sound processing circuit will generate the according melody synthesis or percussion upon receiving the control code from the μC. These signals in turn activate an external amplifier through a current type of D/A converter of a melody output. Melodious sounds are thus generated.

The HT3630 is ideally for applications on advanced multi-key pianos, musical synthesizers, etc.

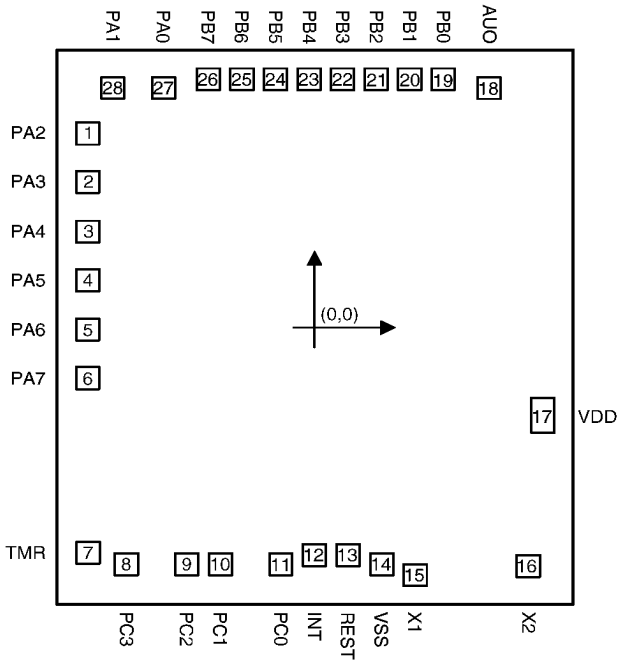
Pin Assignment



Block Diagram



Pad Coordinates



Chip size: $2730 \times 3030\ (\mu m)^2$

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

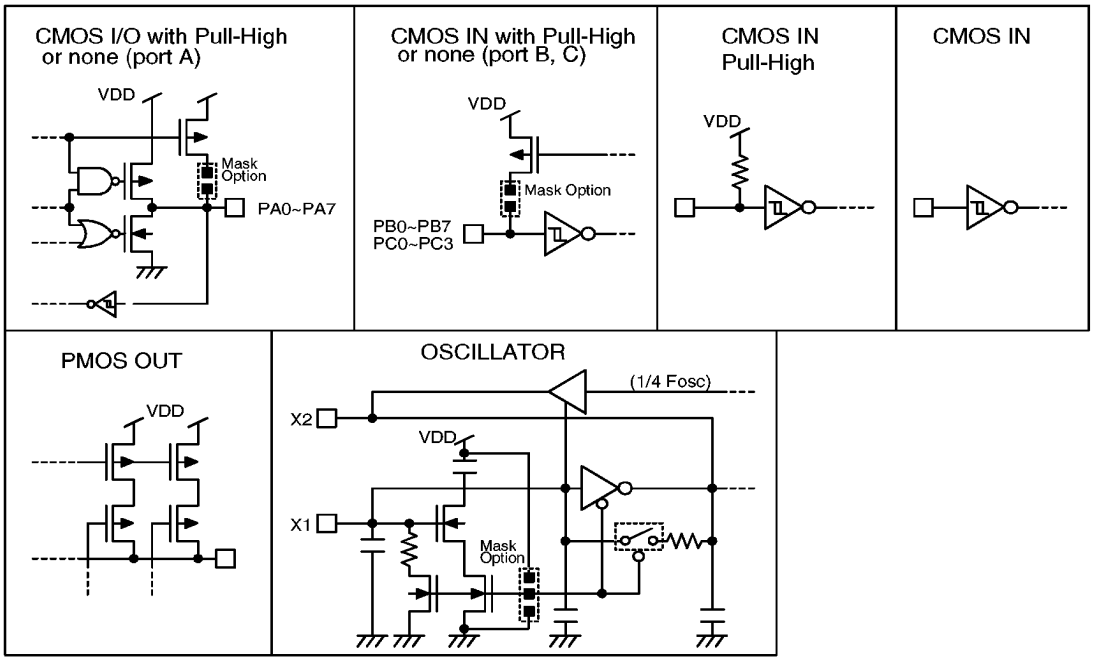
Unit: μm

Pad No.	X	Y	Pad No.	X	Y
1	–1183	1056.8	15	526.2	–1353.3
2	–1183	792.2	16	1115.5	–1303.6
3	–1183	521.4	17	1191.5	–517
4	–1183	256.8	18	911.8	1304.5
5	–1183	–14	19	671.5	1352.6
6	–1183	–278.6	20	496.5	1352.6
7	–1180	–1231	21	321.5	1352.6
8	–980.8	–1293.3	22	146.5	1352.6
9	–666.3	–1293.3	23	–28.5	1352.6
10	–489.9	–1293.3	24	–203.5	1352.6
11	–175.4	–1293.3	25	–378.5	1352.6
12	1.2	–1243.2	26	–553.5	1352.6
13	176.2	–1243.2	27	–788.2	1305.6
14	353	–1293.3	28	–1052.8	1305.6

Pad Description

Pad No.	Pad Name	I/O	Internal Connection	Description
27,28 1~6	PA0~PA7	I/O	CMOS with Pull-High or None	Bidirectional 8 bit input/output port The port address is 12H. Each bit can be configured as a wake-up input by mask option. Software instructions determine a CMOS output or a schmitt trigger input with or without a pull-high resistor (by mask option).
7	TMR	I	CMOS	Schmitt trigger input for a timer/event counter
8~11	PC3~PC0	I	CMOS with Pull-High or None	4 bit input lines Schmitt trigger input with or without a pull high resister by mask option The port address is 16H.
12	INT	I	CMOS Pull-High	External interrupt schmitt trigger input with a pull high resistor Edge trigger is activated on a high to low transition.
13	REST	I	CMOS	System reset input pad for the μC, active low
14	VSS	I	—	Power supply (negative)
15,16	X1,X2	I,O	—	The X1 and X2 are connected to an RC network or a crystal (by mask option) for an internal system clock. In the case of RC operation, the X2 is the output terminal of a 1/4 system clock.
17	VDD	I	—	Power supply (positive)
18	AUO	O	PMOS	Audio signal output The AUO output is a current type D/A. The outputs of the melody channels 1~2 and the output of the percussion are all from the AUO. It drive a power amplifier or a transistor for applications.
19~26	PB0~PB7	I	CMOS with Pull-High or None	8 bit input port The port address is 14H. Schmitt trigger input with or without a pull high resistor (by mask option).

Approximate internal connection circuits



Absolute Maximum Ratings*

Supply Voltage ..... –0.3V to 5.5V
Storage Temperature ..... –50°C to 125°C
Input Voltage ..... V_{SS}–0.3V to V_{DD}+0.3
Operating Temperature ..... 0°C to 70°C

*Note: Stresses above those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. These are stress ratings only. Functional operation of this device at these or any other conditions above those indicated in the operational sections of this specification is not implied and exposure to absolute maximum rating conditions for extended periods may affect device reliability.

Electrical Characteristics

(Ta=25°C)

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
V_{DD}	Operating Voltage	—	—	3.3	4.5	5.1	V
I_{DD}	Operating Current	4.5V	No load, F_{OSC}=3.58MHz	—	3.5	7.0	mA
I_{STB}	Standby Current	4.5V	No load, System halt	—	1	5	μA

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
I_{OH}	Output Source Current (PA0~PA7)	4.5V	V_{OH}=4.05V	–2.7	–4.6	—	mA
I_{OL}	Output Sink Current (PA0~PA7)	4.5V	V_{OL}=0.45V	3.4	5.6	—	mA
I_{OUT}	Max. AUO Output Current	4.5V	V_{OUT}=0.6V	–2	–3	–5.8	mA
I_{IL}	Input Current (for ports A,B,C with a pull-high resistor)	4.5V	V_{IL}=0V	50	70	105	μA
V_{IH}	Input High Voltage for Input Port	—	—	0.8V_{DD}	—	V_{DD}	V
V_{IL}	Input Low Voltage for Input Port	—	—	V_{SS}	—	0.2V_{DD}	V
F_{OSC}	System Frequency	4.5V	3.58MHz crystal or 91kΩ	—	—	—	MHz
t_{WDTOSC}	Watchdog Oscillator	—	—	31	78	140	μs
t_{WDT1}	Watchdog Time-out Period (RC)	—	Without a WDT prescaler	8	20	36	ms
t_{WDT2}	Watchdog Time-out Period (system clock)	—	Without a WDT prescaler	—	1024	—	t_{SYS}
t_{RES}	External Reset Low Pulse Width	—	—	1	—	—	μs
t_{INT}	Interrupt Pulse Width	—	—	1	—	—	μs

* Notes: t_{SYS}=1/(f_{SYS})

ETS & Percussion

The μC (microcontroller) of the HT3630 controls 2 channels of ETS (Electronic Tone Synthesizer) as well as a percussion generator through control registers (refer to the block diagram).

The ETS circuit deals with the data of a control register issued by the commands of the μC. It outputs signals of timbres, rhythms, volume and various scales of sounds defined by users after receiving the data of the control register. The ETS is comprised by a waveform PLA along with an envelope generator. The waveform PLA which is programmed with basic waveforms together with the envelope generator generate various kinds of timbres. Scale of a timbre is decided by the scale generator which is also programmable through the control registers.

The percussion generator, on the other hand, supplies signals of percussion or various tones. The Bass drum, High hat, Cow bell, Snare drum and cymbal can be simulated internally and controlled by PRCU register.

The output of melody channels and percussion channel are all from AUD pin which drive a speaker through an external power amplifier.

Function of Control Registers

Register Name	Function	Control bits 7	6	5	4	3	2	1	0
DIVL DIVH	Scale frequency decision		DV6	DV5	DV4	DV3	DV2 DV9	DV1 DV8	DV0 DV7
DIVH	Octave selection			OC2	OC1	OC0			
	Tone output control		KON						
WAVE	Sustain level selection						SL2	SL1	SL0
	Waveform selection			WV2	WV1	WV0			
ADCK	Attack rate control			AT2	AT1	AT0			
	Decay rate control						DC2	DC1	DC0
SRCK	Sustain rate control			SU2	SU1	SU0			
	Release rate control						RL2	RL1	RL0
PRCU	Percussion selection				PR4	PR3	PR2	PR1	PR0
	Volume selection		MV1	MV0					
HALT	Halt ETS (write a non-zero data in this register)								

Address of control registers & usage

Please refer to the table listed below and according to the following seriation to write the data onto the ETS controller. First give a register address of ETS, and then write a data into the register. Notice the interval between the commands that address the registers of ETS should be $64/F_{SYS}$ at least. Following is a table illustrating the correspondence between control registers and their bits.

Address=1EH (Register Address of ETS)								Address=1FH (Control Data of ETS)								Register Name
7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	
x	x	x	0	0	0	x	@	—	DV6	DV5	DV4	DV3	DV2	DV1	DV0	DIVL
x	x	x	0	0	1	x	@	—	KON	OC2	OC1	OC0	DV9	DV8	DV7	DIVH
x	x	x	0	1	0	x	@	—	—	WV2	WV1	WV0	SL2	SL1	SL0	WAVE
x	x	x	0	1	1	x	@	—	—	AT2	AT1	AT0	DC2	DC1	DC0	ADCK
x	x	x	1	0	0	x	@	—	—	SU2	SU1	SU0	RL2	RL1	RL0	SRCK
x	x	x	1	0	1	x	x	—	MV1	MV0	PR4	PR3	PR2	PR1	PR0	PRCU
x	x	x	1	1	0	x	x	—	—	—	—	—	—	—	—	HALT
x	x	x	1	1	1	x	@	—	—	—	—	—	—	—	—	TEST

Notes: 1. "—" denotes "inapplicable".

2. "x" is the don't care status. It can be "1" or "0".

3. "@" (bit0 of the ETS register address) denotes the channel number (channels 1~2). When @=0, it means active channel is 1. When @=1, it means active channel is 2.

4. The TEST register is reserved for use in the chip testing, it is not user applicable.

Tone (scale) generator

- Timbre

  One cycle of the simulated instrument waveform which is called timbre are recorded in the wave PLA, the PLA can record 8 kind of timbres at maximum.

- Scales

  Different scales can be obtained from changing the retrieval speed of PLA data, the retrieval speed is equal to the output frequency of the tone (scale) generator.

  The frequency of a tone is decided by the data of DV0~DV9. It is derived by a division operation of the system frequency by DV0~DV9. DV6~DV0 of the DIVL register have to be sent first, followed by DV9~DV7 (bits 2~0) of the DIVH register in the process of sending data. The data of DIVL and DIVH have to be re-sent each time the tone frequency is altered.

- Octave

  OC0~OC2 of the DIVH register selecte the octave value, ranging from 0~4 (the binary value 5~7 are not used).

- Scale formula

  The formula of a tone frequency is:

$$F_{OUT} = \frac{F_{OSC}}{DIV \times 2^{(7-OCT)}}$$

  where F_{OUT} is the output signal frequency, F_{OSC} the system operation frequency, DIV of 10 bits DV9~DV0 the decade number and OCT the octave (0~4).

  Notice that the values of DV9 and DV8 cannot be both zero.

  For example, if F_{OSC} is 3.58MHz, DIV is 357H (equal to a decade number 855) and OCT is 3H,

$$F_{OUT} = \frac{3.58 \times 10^6}{357H \times 2^{(7-3)}} = \frac{3.58 \times 10^6}{855 \times 2^{(7-3)}} = 261.6Hz \text{ (center C)}$$

Scale table

(The values of DIVL and DIVH are both hexadecimal.)

Compass	DIVL (DV7~DV0)	DIVH (DV9~DV8)
C	57	06
C#	27	06
D	7A	05
D#	4F	05
E	27	05
F	01	05
F#	5D	04
G	3B	04
G#	1B	04
A	7C	03
A#	60	03
B	45	03

Envelope programming

- Envelope of a waveform

  The envelope of a timbre can be divided into four parts, namely A, D, S and R (A=Attack, D=Decay, S=Sustain, R=Release). Its waveform is shown below:

- A, D, S, R

  Envelope will first rise from the bottom level with a designated attack rate to the top level and then fall down by decay rate to the designated sustain level when KON=1. After that, it will go on falling down by sustain rate to the bottom level. If KON=0 before reaching the bottom level, envelope will keep falling down at a releasing rate till it reaches the bottom level.

- Defining A, D, S, R

  The rates of A, D, S and R can be user defined (by the ADCK and SRCK registers). Sustain can be further divided into 8 levels (part of top level) which are defined by users (by the SL2~SL0 bits of the WAVE register).

Value (hexadecimal)	Time from bottom level to top level (ms)				Sustain level
	Attack	Decay	Sustain	Release	
7	1.2	73.2	∞	73.2	1/8
6	4.8	146.4	146.4	146.4	2/8
5	18.3	292.9	292.9	292.9	3/8
4	36.6	585.8	585.8	585.8	4/8
3	73.2	1171.6	1171.6	1171.6	5/8
2	146.4	2343.2	2343.2	2343.2	6/8
1	292.9	4686.4	4686.4	4686.4	7/8
0	585.8	9372.8	9372.8	9372.8	1

A, D, S and R rise or fall speeds

* Notes: The level of the sustain is from the bottom to the top level.

Wave selection

- Timbre selection

  WV2~WV0 of the WAVE register selects a waveform for a tone (i.e, a timbre).

  Following is a table illustrating the values of the WAVE, ADCK and SRCK registers of built-in timbres.

Timbre	Register		
	WAVE	ADCK	SRCK
Piano	04	3F	15
Organ	0F	3C	24
Guitar	12	1E	3E
Banjo	1E	2E	3E
Flute	22	1E	3E
Trumpet	2C	3E	1B
Saxophone	32	1E	3E
Ebass	3C	3F	13

- Waveform format

  The total amount of waveforms in a waveform memory of 512×7 bits is 8. Therefor, the recording space for each waveform is 64 samples and the resolution of amplitude for a sample is 1/128, i.e, 7-bit PCM coding.

Percussion generation

The percussion circuit controls the sound output selected by the PRCU register. Percussion can consist of a maximum of 4 instruments. The High-hat and the Cymbal instruments, however, cannot be selected at the same time.

PRCU register					Instrument
PR4	PR3	PR2	PR1	PR0	
0	0	0	0	1	Bass drum
0	0	0	1	0	High hat
0	0	1	0	0	Cow bell
0	1	0	0	0	Snare drum
1	0	0	1	0	Cymbal

Volume control

The HT3630 provides 4 different volume control modes by writing data to the bits MV1 and MV0 of the PRCU register. The following table illustrates the proportion of the melody channel and the sound channel.

PRCU register		Volume (the ratio of the maximal output)		
		Melody channel		Sound channel
MV1	MV0	0	1	
0	0	1/4	1/4	1/2
0	1	0	0	1
1	0	1/2	1/2	0
1	1	1	0	0

System halt

The ETS is suspended whenever a non-zero code is writen into the HALT register.

It is required to halt the ETS before the whole system goes into HALT.

Test

The TEST register only for use in testing the ETS circuit. No data can be written to the TEST register for normal operations.

MicroController

The HT3630 is built-in with an 8 bit high performance RISC-like μC (microcontroller). The μC scans a keyboard and processes key trigger inputs. It can design special functions, such as auto-accompaniment, auto-harmony and transpose.

The μC includes a program ROM of 2K×14 bits in addition to a data RAM of 96×8 bits. The ROM can be programmed to scan a keyboard whereas the RAM memorizes and stores tones.

Following is a detailed description on the μC.

Program counter - PC

The 11 bit program counter (PC) controls the sequence in which the instructions stored in the program ROM are executed. It can specify a maximum of 2048 addresses.

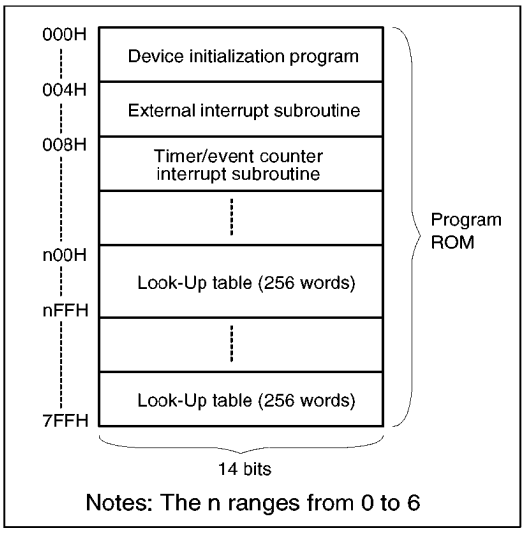
The lower byte of the program counter (PCL) is a readable and writeable register (06H). Moving data into PCL performs a short jump. Its destination is within 256 locations.

Program memory - ROM

The program memory stores the to-be-executed program instructions. It can also contains data, table as well as interrupt entries. The memory is organized with 2048×14 bits, addressed by the program counter along with a table pointer. Certain locations in the program memory are reserved for special usages.

- Location 000H

  This area is reserved for program initialization. The program begins execution always at location 000H after a chip reset.



Program memory

Mode	Program Counter										
	*10	*9	*8	*7	*6	*5	*4	*3	*2	*1	*0
Initial reset	0	0	0	0	0	0	0	0	0	0	0
External interrupt	0	0	0	0	0	0	0	0	1	0	0
timer counter overflow	0	0	0	0	0	0	0	1	0	0	0
Skip	PC+2										
Loading PCL	*10	*9	*8	@7	@6	@5	@4	@3	@2	@1	@0
Jump, call branch	#10	#9	#8	#7	#6	#5	#4	#3	#2	#1	#0
Return from subroutine	S10	S9	S8	S7	S6	S5	S4	S3	S2	S1	S0

Porgram counter

Notes: *10~*0: Bits of program counter
S10~S0: Bits of stack register
#10~#0: Bits of instruction code
@7~@0: Bits of PCL

- Location 004H

  This area is reserved for an external interrupt service program. The program begins execution at location 004H if the INT input pin is activated under the conditiion that interrupt is enabled and the stack is not full.

- Location 008H

  This area is reserved for a timer interrupt service program. The program begins execution at location "008H" if a timer interrupt results from a timer overflow, under the condition that interrupt is enabled and the stack is not full.

- Table location

  Any location in the ROM can be used as a look–up table. The instructions "TABRDC [m]" (the current page, 1 page=256 words) and "TABRDL [m]" (the last page) transfer the contents of the lower-order byte to the specified data memory, and the higher-order byte to TBLH (08H). Only the destination of the lower-order byte in the table is well-defined; the other bits of the table word are transferred to the lower portion of TBLH. The remaining bit is read as "0". The table higher-order byte register (TBLH) is read only. The table pointer (TBLP) is a read/write register (07H) indicating the table location. The location has to be placed in TBLP before accessing the table. All of the table related instructions require 2 cycles to complete an operation. These areas may function as a normal program memory depending upon the user's requirements.

Stack register - STACK

The stack register is a special part of memory used to save the contents of the program counter (PC), when a program jump occrs. The stack is organized into 2 levels. It is neither part of the data nor program, and is neither readable nor writeable. The activated level is indexed by the stack pointer (SP) which is neither readable nor writeable. SP will point to the top of the stack after a chip reset.

- Driven by interrupt

  If the stack is full and a non-masked interrupt takes place, an interrupt request flag will be recorded but the acknowledgment be inhibited. The interrupt will be serviced to prevent a stack overflow after the stack pointer is decremented (by RET or RETI), which allowing the programmer to use the structure more easily.

- Driven by instructions

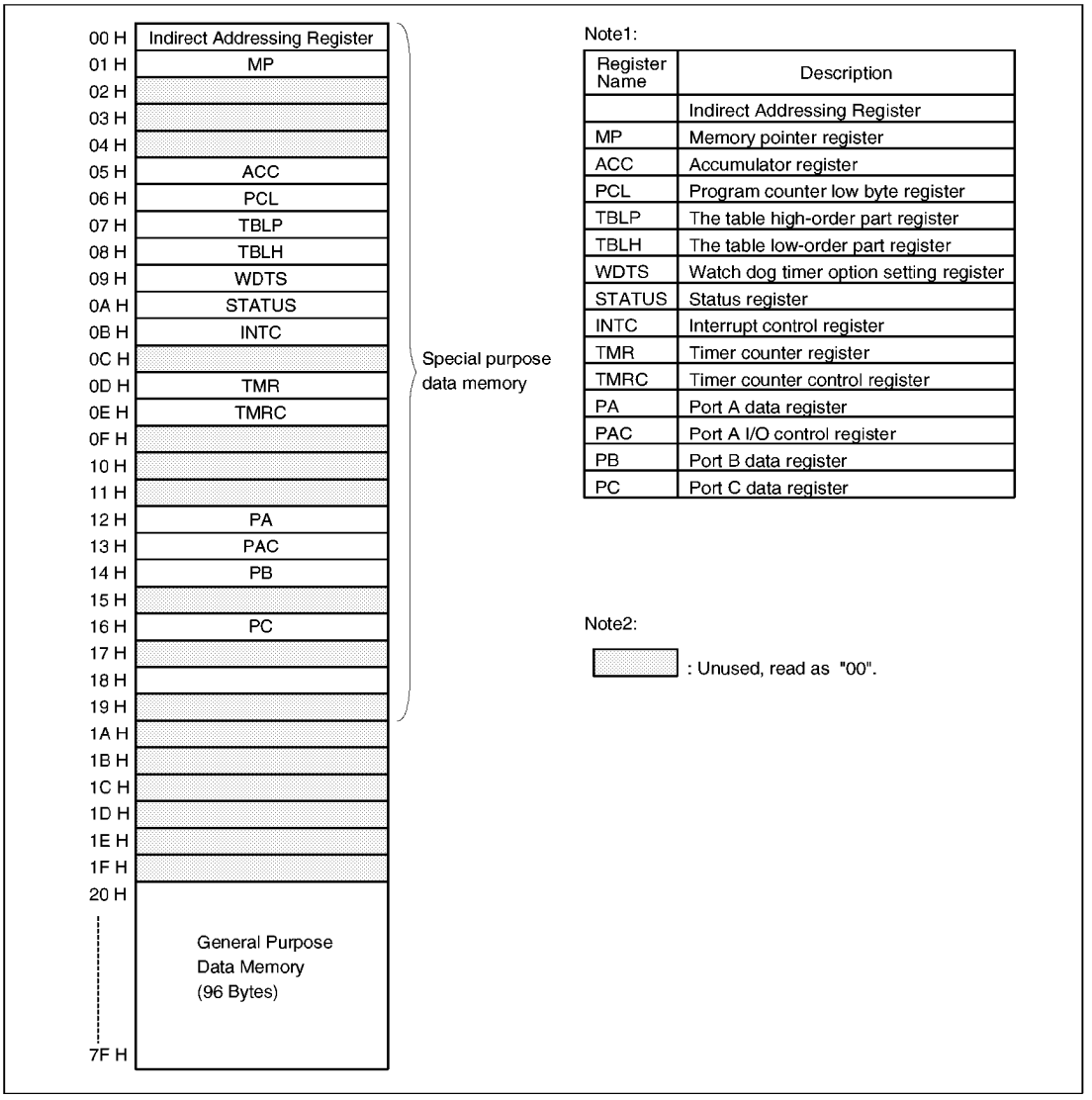
  On the other hand, if the stack is full and a "CALL" is subsequently executed, a stack overflow occurs and the first entry is lost (only the most two recent return addresses are stored).

Data memory - RAM

The data memory is designed with 112×8 bits. It is divided into two functional groups, namely special function registers and general purpose data memory. Although most of them are read/write, some are read only.

Instruction(s)	Table Location										
	*10	*9	*8	*7	*6	*5	*4	*3	*2	*1	*0
TABRDC[m]	P10	P9	P8	@7	@6	@5	@4	@3	@2	@1	@0
TABRDL[m]	1	1	1	@7	@6	@5	@4	@3	@2	@1	@0

Notes: *10~*0: Bits of table location
@7~@0: Bits of table pointer
P10~P8: Bits of current program counter



RAM Mapping

- Special function registers

  The special function registers are from 00H to 16H. The remaining space before 20H is reserved for future expansion. Reading these locations will get "00H".

- General purpose registers

  The general purpose data memory, addressed from 20H to 7FH, is used for storing data and control information that are generated or reguired by instruction commands.

  Each of the data memories can implement arithmetic, logic, increment, decrement and rotate operations directly. Each bit of the data memory can be set and reset by the "SET

[m].i" and "CLR [m].i" instructions except some dedicated bits. It is also indirectly accessible through a memory pointer register (MP;01H).

Indirect addressing register

Location 00H is an indirect addressing register not physically implemented. Any read/write operation of [00H] accesses the data memory that is pointed to by MP (01H). Reading location 00H indirectly will return the result 00H but writing it indirectly will lead to no operations.

Accumulator

The accumulator is closely relates to the ALU operations. It is also mapped to location 05H of the data memory and capable of operating with immediate data. The data movement between two data memories must get through the accumulator.

Arithmetic and logic unit - ALU

This circuit performs 8 bit arithmetic and logic operations. ALU provides the following functions:

- Arithmetic operations (ADD, ADC, SUB, SBC, DAA)
- Logic operations (AND, OR, XOR, CPL)
- Rotation (RL, RR, RLC, RRC)
- Increment & Decrement (INC, DEC)
- Branch decision (SZ, SNZ, SIZ, SDZ ....)

ALU not only stores the results of a data operation but also change the status register.

Status register - STATUS

The 8 bit register (0AH) consists of a zero flag (Z), carry flag (C), auxiliary carry flag (AC), overflow flag (OV), power down flag (PD) and watch dog time-out flag (TO). It also records the status information in addition to controlling the operation sequence.

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

Like other registers, bits in the status register can be altered by instructions except the TO and PD flags.

Labels	Bits	Function
C	0	C is set if an operation results in a carry during an addition operation or if a borrow doesn't take place during a subtraction operation; otherwise C is cleared. It is also affected by a rotate through carry instruction.
AC	1	AC is set if an operation results in a carry out of the low nibbles in addition or if no borrow takes place from the high nibble into the low nibble in subtraction; otherwise it is cleared.
Z	2	Z is set if the result of an arithmetic or logic operation is zero; otherwise it is cleared.
OV	3	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 it is cleared.
PD	4	PD is cleared when the system powers up or a WDT overflow occurs during a normal operation. It is set by executing the "HALT" instruction.
TO	5	TO is cleared by a system power-up or executing the "CLR WDT" or "HALT" instruction. TO is set by a WDT time-out.
–	6	Undefined, read as "0".
–	7	Undefined, read as "0".

Status register

- Precaution

  Operations related to the status register may derive different results from those intended. For example, clearing the status register CLR [0AH] imposes no effects on flags TO and PD. The zero flag is 1 also. uu0100 is then the data of the register, where "u" means unchanged value.

  The status register will not be automatically pushed onto the stack when entering the interrupt sequence or executing the subroutine call. If its contents are important and the subroutine can corrupt the status register, the programmer should take precautions to save it properly.

- TO & PD

  Any data written to the status register will not change the TO or PD flags.

  The TO and PD flag can be changed by the WDT overflow, chip power-up, clearing WDT or by "HALT" instruction.

Interrupt

The μC of the HT3630 provides an external interrupt as well as internal timer interrupts. The interrupt control register (INTC;0BH) includes interrupt request flags as well as interrupt control bits to set the status of enable/disable.

- INTC contents

  The timer interrupt request flag (TF), external interrupt request flag (EIF), enable timer bit (ETI), enable external interrupt bit (EEI) and enable master interrupt bit (EMI) constitute an interrupt control register (INTC) which is located at 0BH of the data memory.

- Interrupt nesting

  Once an interrupt subroutine is serviced, other interrupts are all blocked (by clearing the EMI bit). This scheme may prevent any further interrupt nesting. Other interrupt requests may occur during this interval but only the interrupt request flag is recorded. If an interrupt requires servicing within the service routine, the programmer may set the EMI bit and the corresponding bit of INTC, allowing interrupt nesting. If the stack is full, the interrupt request will not be acknowledged until SP is decremented even though its related INTC bits is enabled. The stack should be prevented from being full if immediate service is desired.

- Wake-up

  All of the interrupts have a wake-up capability.

Register	Bit No.	Label	Function
INTC (0BH)	0	EMI	Controls a master (global) interrupt. (1=enabled; 0=disabled)
	1	EEI	Controls an external interrupt. (1=enabled; 0=disabled)
	2	ETI	Controls a timer interrupt. (1=enabled; 0=disabled)
	3	–	Unused bit, read as "0".
	4	EIF	External interrupt request flag (1=active; 0=inactive)
	5	TF	Internal timer request flag (1=active; 0=inactive)
	6	–	Unused bit, read as "0".
	7	–	Unused bit, read as "0".

INTC register

- Data retrieval

  As an interrupt is serviced, a control transfer occurs by pushing the program counter onto the stack and then branching to subroutines at specified location(s) of the program memory. Only the contents of program counter can be pushed onto the stack. If the contents of the register or status register (STATUS) are altered by an interrupt service program which corrupts the desired control sequence, the programmer must save these contents in advance.

- External interrupt

  External interrupts are triggered by a high to low transition of INT and the related interrupt request flag (EIF; bit 4 of INTC) is set accordingly. A subroutine call to location 004H and EMI will occur when external interrupt occurs during EEI enable if the stack is not full. At the same time, the external interrupt request flag (EIF) and EMI bits are both cleared to disable other interrupts.

- Internal interrupt

  The internal timer/counter interrupt is initialized by set of the timer/counter interrupt request flag (TF; bit 5 of INTC), which results from a timer overflow. A subroutine call to location 008H will occur when the TF is set during EMI and ETI enable if the stack is not full. At the same time its related interrupt request flag (TF) will be reset and the EMI bit cleared to disable further interrupts.

  During the execution of an interrupt subroutine, other interrupt acknowledgments are all held until the "RETI" instruction is executed or the EMI bit and related interrupt control bit are set to 1 (if the stack is not full).

- RETI

  To return from the interrupt subroutine, "RET" or "RETI" may be invoked. RETI will set the EMI bit to enable an interrupt service, but RET will not.

- Interrupt priority

  Interrupts which occur at an interval between the rising edges of two consecutive T2 pulses will be serviced at the latter T2 pulse when the corresponding INTC bits are enabled. In the case of simultaneous requests priorty of interrupt source is listed in the following table:

No.	Interrupt Source	Priority	Vector
a	External interrupt	1	004H
b	timer overflow	2	008H

- Interrupt enable/disable

  EMI, EEI, ETI control the enable/disable status of interrupts. Reset of these bits prevent the requested interrupt from being serviced.

- Interrupt request

  Once the interrupt request flags (TF, EIF) are set, they will remain in the INTC register until the interrupts are serviced or cleared by a software instruction.

- Precaution

  A program is suggested not to use the "CALL subroutine" within the interrupt subroutine. It's because interrupts often occur in an unpredictable manner or demands to be serviced immediately in some applications. If only one stack is left and the interrupt is not well controlled after being enabled, once the "call" subroutine operates in the interrupt subroutine, the original control sequence is damaged.

Oscillator configuration

- Mask option for RC/X'tal

  The HT3630 includes 2 oscillator circuits designed for system clocks, namely RC oscillator and crystal oscillator, which are determined by mask option.

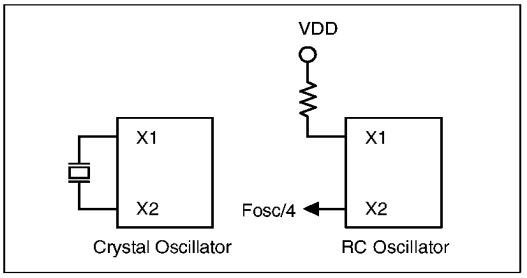
- HALT

  The HALT mode stops the system oscillator and resists external signal to conserve power.

- RC oscillator

  If an RC oscillator is used, an external resistor between X1 and VDD is required. The system clock, divided by 4, is available on X2, and can be used to synchronize external logic. The RC

oscillator provides a most cost effective solution, but the frequency of the oscillation may vary with VDD, temperature and the chip itself due to process variations. It is, therefore, not suitable for timing sensitive operations where an accurate oscillator frequency is desired.



System Oscillator

- X'tal oscillator

  If the crystal oscillator is used, a crystal across X1 and X2 is required to provide the feedback and phase shift needed for an oscillator. No other external components are required. A resonator can replace a crystal oscillator to be connected between X1 and X2, yielding a frequency reference. But two external capacitors in X1 and X2 are required.

- WDT oscillator

  The WDT oscillator is a free running on-chip RC oscillator. No external components are required. The WDT oscillator still active with an oscillating period of approximately 78us even if the system enters the power down mode in which the system oscillator stops. The WDT oscillator can be disabled by mask option to conserve power.

Watch dog timer - WDT

- Mask option
  * WDT source

    The clock source of WDT is implemented by an dedicated RC oscillator (WDT oscillator) or instruction clock (system clock divided by 4), decided by mask option.

  * WDT disable

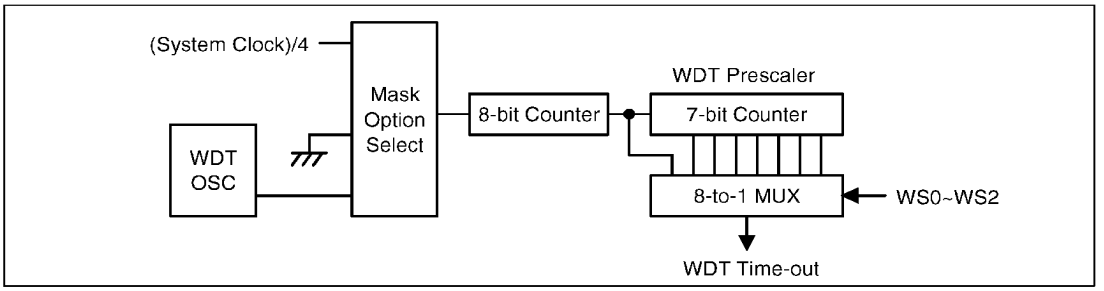
    This timer prevents a software malfunction or sequence from jumping to an unknown location with unpredictable results. It can be disabled by mask option. All the executions related to WDT lead to no operation if the WDT timer is disabled.

- RC osc option

  Once an internal WDT oscillator (RC oscillator whose period is 78μs normally) is selected, it is first divided by 256 (8 stages) to get a nominal time-out period of approximately 20ms. This time-out period may vary with temperature, VDD and process variations.

- WDTS

  Longer time-out periods can be realized by invoking the WDT prescaler. Writing data to WS2, WS1 and WS0 (bits 2,1,0 of WDTS–WDT setting register) can derive different time-out periods. If WS2, WS1 and WS0 all equal to 1, the division ratio is up to 1:128 and the maximum time-out period is about 2.6 seconds.



Watch dog timer

- Instruction clock option

  If the WDT oscillator is disabled, the WDT clock can still be fed from the instruction clock and operates in the same manner. Nonetheless, the WDT not only stops counting but loses its protecting purpose when in the HALT state. In this situation the logic can only be re-initialized by external logic.

- User flags

  The high nibble and bit 3 of WDTS are reserved for user defined flags. The programmer may use these flags to indicate some specified status.

- Notice

  If the device operates in a noisy environment, using an on-chip RC oscillator (WDT OSC) is strongly recommended, since HALT will terminate the system clock.

WS2	WS1	WS0	Division Ratio
0	0	0	1:1
0	0	1	1:2
0	1	0	1:4
0	1	1	1:8
1	0	0	1:16
1	0	1	1:32
1	1	0	1:64
1	1	1	1:128

- WDT overflow

  An overflow of WDT will initialize a "chip reset" and set the status bit "TO" under normal operations. It, on the other hand, will initialize a "warm reset" and only PC and SP be reset to zero in the HALT mode.

- Clear WDT

  To clear the contents of WDT (including the WDT prescaler), 3 methods can be adopted, namely external reset (a low level to REST), software instruction(s) and "HALT" instruction.

- CLR WDT instructions

  The software instruction(s) includes "CLR WDT" and the other sets – "CLR WDT1" and "CLR WDT2". Of these two types of instruction, only one instruction can be active at a time by mask option – "CLR WDT times selection option". Any execution of the "CLR WDT" instruction will clear WDT if CLRWDT times equal one. The "CLR WDT1" and "CLR WDT2" instructions have to be executed to clear WDT if CLRWDT times equal two. Otherwise, WDT may reset the chip as a result of time-out.

Power down operation - HALT

The HALT mode is initialized by the "HALT" instruction and results in the following.

- The system oscillator is turned off but the WDT oscillator keeps running (if the WDT oscillator is selected).
- The contents of on–chip RAM and registers remain unchanged.
- The WDT and WDT prescaler will be cleared and recount again (if the clock of WDT is from the WDT oscillator).
- All I/O ports remain their status before HATL.
- The PD flag is set and the TO flag is cleared.
- Quit HALT

  The system can quit the HALT mode by an external reset, interrupt, external falling edge signal on port A or WDT overflow. An external reset causes device initialization and a WDT overflow performs a "warm reset".

- Cause for reset

  The reason for chip reset can be determined after examining the TO and PD flags.

- PD flag

  The PD flag is cleared when the system powers up or executes the "CLR WDT" instruction. It is set after the "HALT" instruction is executed.

- TO flag

  The TO flag is set if a WDT time-out occurs, and causes a wake-up that resets only PC and SP. The others maintain their original status.

- Wake-up

  The port A wake-up and interrupt methods can be considered as a continuation of a normal execution.

- Port A

  Each bit of port A can be independently selected to wake up the device by mask option. The program will resume execution of the next instruction after awakening from an I/O port stimulus.

- Interrupt

  Two sequences, on the other hand, may happen when the program awakens from an interrupt. The program will resume execution at the next instruction if the related interrupt(s) is (are) disabled or the interrupt(s) is enabled but the stack is full. A regular interrupt response takes place if the interrupt is enabled and the stack is not full.

- Resuming delay

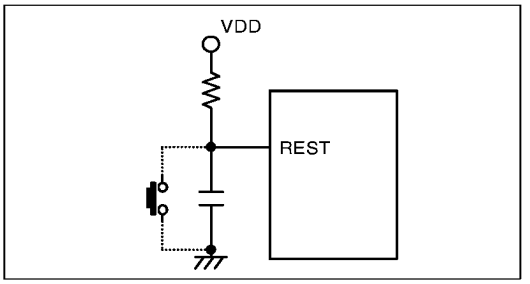
  The μC will takes 1024 t_{SYS} (system clock period) to resume a normal operation after the wake-up event(s) occurs, if the system clock is from a crystal oscillator. That is to say, the μC will insert a dummy period after wake-up. It, on the other hand, will continue operation without any delay if the system clock is from an RC oscillator. The actual interrupt subroutine execution will be delayed by one cycle if wake-up results from an interrupt acknowledgment. The μC will start execution after finish of a dummy period if the wake-up results in the next instruction execution.

- Notice

  All I/O pins should be carefully managed before entering the HALT status to minimize power consumption.

Reset

Most registers are reset to an "initial state" when the reset conditions are satisfied. But some registers remain unchanged under certain reset conditions.



Reset Circuit

There are 3 ways in which a reset can occur:

- REST reset during a normal operation
- REST reset during HALT
- WDT time-out reset during a normal operation

The WDT time-out during HALT is different from other chip reset conditions, since it can perform a "warm reset" that resets PC and SP and leaves other circuits at their current state.

- Reset interpretation

  The program can distinguish between different "chip resets" by examining the PD flag and TO flag as shown.

TO	PD	RESET Conditions
0	0	REST reset during power-up
u	u	REST reset during a normal operation
0	1	REST wake-up HALT
1	u	WDT time-out during a normal operation
1	1	WDT wake-up HALT

* Notes: "u" means "unchanged".

The states of the registers are summarized in the following table:

Register	Reset (power on)	WDT time-out (normal operation)	REST reset (normal operation)	REST reset (HALT)	WDT time-out (HALT)
TMR	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu
TMRC	00-0 1---	00-0 1---	00-0 1---	00-0 1---	uu-u u---
PC	000H	000H	000H	000H	000H*
MP	-xxx xxxx	-uuu uuuu	-uuu uuuu	-uuu uuuu	-uuu uuuu
ACC	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu
TBLP	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu
TBLH	--xx xxxx	--uu uuuu	--uu uuuu	--uu uuuu	--uu uuuu
STATUS	--00 xxxx	--1u uuuu	--uu uuuu	--01 uuuu	--11 uuuu
INTC	--00 -000	--00 -000	--00 -000	--00 -000	--uu -uuu
WDTS	0000 0111	0000 0111	0000 0111	0000 0111	uuuu 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
PC	---- 1111	---- 1111	---- 1111	---- 1111	---- uuuu

Notes: "*" means a "warm reset".
"u" means "unchanged".
"x" means "unknown".

- Start execution

  The μC has another useful feature for testing and synchronization purposes. Releasing REST high will begin execution if REST keeps low long enough.

- Reset status

  The chip reset status of the functional units are shown below.

PC	000H
Interrupt	Disable
Prescaler	Clear
WDT	Clear; after master reset, WDT begins counting.
Timer	Off
Input/output Ports	Input mode
SP	Point to the top of a stack

Timer/counter

The timer/counter contains an 8 bit programmable count-up counter. The clock is from the system clock divided by 4.

- Timer register

  Two registers are related to the timer; TMR ([0DH]), TMRC ([0EH]) and two physical registers mapped to the TMR location. Writing TMR makes the starting value put in the timer preload register and reading it gets the contents of the timer. TMRC is a timer control register defining some options.

- Timer mode

  The default timer mode (TM0=0, TM1=1) functions as a normal timer with the clock source from the instruction clock.

- Overflow

  Once the counter starts counting, it will count from its current value to FFH. It is reloaded from the timer/counter preload register and generates an interrupt request flag (TF; bit 5 of INTC) when an overflow occurs.

- Timer enable/disable

  TON can be reset only by instructions. The overflow of the timer/counter is one of the wake-up sources.

Label (TMRC)	Bits	Function
–	0~2	Unused bits, read as “0”.
TE	3	To define the TMR active edge of the timer (0=active on low to high; 1=active on high to low)
TON	4	To enable/disable timer counting (0=disabled; 1=enabled)
–	5	Unused bits, read as “0”.
TM0 TM1	6 7	10=Timer mode (internal clock) 00, 01 and 11 are not used.

TMRC register

- Timbre interrupt

  Writing a 0 to ETI of INTL register disables the timer service, no matter what the operation mode is.

- Preload

  Writing data to the timer preload register will also reload that data to the timer in the case of timer/counter off condition. But if the counter is turned on, data written to the counter will be kept only in the timer/counter preload register. The counter will still go on operation until an overflow occurs.

- Reading TMR

  The clock will be blocked to avoid errors when the timer (reading TMR) is read. The action of clock blocking should be taken into consideration since it may results in a counting error.

Input/output ports

The HT3630 provides 8 bi-directional input/output lines (PA) and 12 input lines (PB,PC). The total 20 lines are labeled from PA to PC, and mapped to [12H], [14H] and [16H] of the data memory, respectively. All the I/O ports (PA) can be used as input and output operations.

- Input

  The ports are non-latching in an input operation. They have to be ready at the T2 rising edge of the "MOV A,[m]" (m=12H, 14H and 16H) instruction.

- Output

  All data, on the other hand, are latched and remain unchanged in an output operation till the output latch is rewritten.

- Re-configuration of I/O line

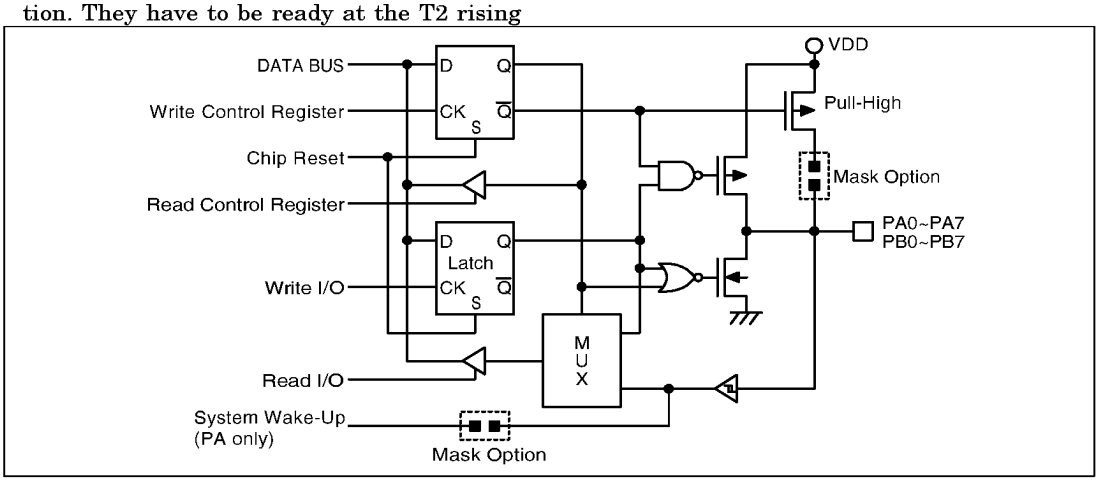
  Each I/O line has its own control register (PAC) to control the input/output configuration. With this control register, a CMOS output or schmitt trigger input with or without pull-high resistor (mask option) structures can be reconfigured dynamically (i.e., on-the-fly) under a software control.

  * As input

    "1" has to be written to the corresponding latch of the control register when it functions as an input. The pull-high resistance will be automatically exhibited if the pull-high option is selected.

  * Input source

    The input source(s) also depend(s) on the control register. An input will read the pad state if the control register bit is "1". The contents of the latches, on the other hand, will move to an internal bus if the control register bit is "0". The latter is possible in an "read-modify-write" instruction.



Input/output ports

* As output
  CMOS is the only configuration for an output function. These control registers are mapped to locations 13H (PAC)

- Reset
  The input/output lines stay at a high level or is floating (by mask option) after a chip reset.
- SET & CLR
  Each bit of the input/output latches can be set or cleared by the "SET [m].i" and "CLR [m].i" (m=12H) instructions.
- Read-modify-write
  Some instructions will first input data and then followed by an output operations. For example, instructions "SET [m].i", "CLR [m].i", "CPL [m]", "CPLA [m]" will read the entire port state into the CPU, execute the defined operations (bit-operation), and then write the results back to the latches or the accumulator.
- Wake-up
  Each line of port A has the capability of waking up the device.

Mask option

The following table illustrates 5 kinds of mask option in the μC. All the mask options have to be defined to ensure proper system functioning.

No.	Mask Option
1	OSC type selection This option decides whether an RC or crystal oscillator is chosen as a system clock.
2	WDT source selection There are 3 types of selection: on-chip RC oscillator, instruction clock and disable WDT.
3	CLRWDT times selection This option defines the way of clearing WDT by instructions. "One time" means "CLR WDT" can clear WDT. "Two times" means WDT can be cleared only after "CLR WDT1" and "CLR WDT2" are both executed. In this case, only "One time" option enable.
4	Wake-up selection This option defines the activity of the wake-up function. External I/O pins (PA only) all have the capability of wakening up the chip from a HALT state.
5	Pull-high selection This option decides whether or not the pull-high resistance is visible in the input mode of the I/O ports & I/P ports. Each bit of an I/O port & I/P port can be independently selected.

Instruction Set

Instruction set summary

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

Mnemonic	Description	Flag Affected
Rotate		
RRA [m]	Rotate data memory right with result in ACC	None
RR [m]	Rotate data memory right	None
RRCA [m]	Rotate data memory right through carry with result in ACC	C
RRC [m]	Rotate data memory right through carry	C
RLA [m]	Rotate data memory left with result in ACC	None
RL [m]	Rotate data memory left	None
RLCA [m]	Rotate data memory left through carry with result in ACC	C
RLC [m]	Rotate data memory left through carry	C
Data Move		
MOV A,[m]	Move data memory to ACC	None**
MOV [m],A	Move ACC to data memory	None
MOV A,x	Move immediate data to ACC	None
Bit Operation		
CLR [m].i	Clear bit of data memory	None
SET [m].i	Set bit of data memory	None
Branch		
JMP addr	Jump unconditional	None
SZ [m]	Skip if data memory is zero	None
SZA [m]	Skip if data memory is zero with data movement to ACC	None
SZ [m].i	Skip if bit i of data memory is zero	None
SNZ [m].i	Skip if bit i of data memory is not zero	None
SIZ [m]	Skip if increment data memory is zero	None
SDZ [m]	Skip if decrement data memory is zero	None
SIZA [m]	Skip if increment data memory is zero with result in ACC	None
SDZA [m]	Skip if decrement data memory is zero with result in ACC	None
CALL addr	Subroutine call	None
RET	Return from subroutine	None
RET A,x	Return from subroutine and load immediate data to ACC	None
RETI	Return from interrupt	None
Table Read		
TABRDC [m]	Read ROM code (current page) to data memory and TBLH	None
TABRDL [m]	Read ROM code (last page) to data memory and TBLH	None

Mnemonic	Description	Flag Affected
Miscellaneous		
NOP	No operation	None
CLR [m]	Clear data memory	None
SET [m]	Set data memory	None
CLR WDT	Clear Watchdog timer	TO,PD
CLR WDT1	Pre-clear Watchdog timer	TO*,PD*
CLR WDT2	Pre-clear Watchdog timer	TO*,PD*
SWAP [m]	Swap nibbles of data memory	None
SWAPA [m]	Swap nibbles of data memory with result in ACC	None
HALT	Enter power down mode	TO,PD

Notes:

x = 8 bits immediate data.

m = 7 bits data memory address

A = accumulator

i = 0...7 number of bits

addr = 11 bits program memory address

√ = Flag(s) is affected

– = Flag(s) is not affected

* = Flag(s) may be affected by the execution status

Instruction Definition

ADC A,[m] Add data memory and carry to accumulator

Description The contents of the specified data memory, accumulator and the carry flag are added simultaneously, leaving the result in the accumulator.

Operation ACC ← ACC+[m]+C

Affected flag(s)

TC2	TC1	TO	PD	OV	Z	AC	C
–	–	–	–	√	√	√	√

ADCM A,[m] Add accumulator and carry to data memory

Description The contents of the specified data memory,"""""" accumulator and the carry flag are added simultaneously, leaving the result in the specified data memory.

Operation [m] ← ACC+[m]+C

Affected flag(s)

TC2	TC1	TO	PD	OV	Z	AC	C
–	–	–	–	√	√	√	√

ADD A,[m] Add data memory to accumulator

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)

TC2	TC1	TO	PD	OV	Z	AC	C
–	–	–	–	√	√	√	√

ADD A,x Add immediate data to accumulator

Description The contents of the accumulator and the specified data are added, leaving the result in the accumulator.

Operation ACC ← ACC+x

Affected flag(s)

TC2	TC1	TO	PD	OV	Z	AC	C
–	–	–	–	√	√	√	√

ADDM A,[m] Add accumulator to data memory

Description The contents of the specified data memory and the accumulator are added. The result is stored in the data memory.

Operation [m] ← ACC+[m]

Affected flag(s)

TC2	TC1	TO	PD	OV	Z	AC	C
–	–	–	–	√	√	√	√

AND A,[m] Logical AND accumulator with data memory

Description Data in the accumulator and the specified data memory performs a bitwise logical_AND operation. The result is stored in the accumulator.

Operation ACC ← ACC "AND" [m]

Affected flag(s)

TC2	TC1	TO	PD	OV	Z	AC	C
–	–	–	–	–	√	–	–

AND A,x Logical AND immediate data to accumulator

Description Data in the accumulator and the specified data performs a bitwise logical_AND operation. The result is stored in the accumulator.

Operation ACC ← ACC "AND" x

Affected flag(s)

TC2	TC1	TO	PD	OV	Z	AC	C
–	–	–	–	–	√	–	–

ANDM A,[m] Logical AND data memory with accumulator

Description Data in the specified data memory and the accumulator performs a bitwise logical_AND operation. The result is stored in the data memory.

Operation [m] ← ACC "AND" [m]

Affected flag(s)

TC2	TC1	TO	PD	OV	Z	AC	C
–	–	–	–	–	√	–	–

CALL addr Subroutine call

Description The instruction unconditionally calls a subroutine located at the indicated address. The program counter increments once to obtain the address of the next instruction, and pushes this onto the stack. The indicated address is then loaded. Program execution continues with the instruction at this address.

Operation Stack ← PC+1
PC ← addr

Affected flag(s)

TC2	TC1	TO	PD	OV	Z	AC	C
–	–	–	–	–	–	–	–

CLR [m] Clear data memory

Description The contents of the specified data memory are cleared to zero.

Operation [m] ← 00H

Affected flag(s)

TC2	TC1	TO	PD	OV	Z	AC	C
–	–	–	–	–	–	–	–

CLR [m].i Clear bit of data memory

Description The bit i of the specified data memory is cleared to zero.

Operation [m].i ← 0

Affected flag(s)

TC2	TC1	TO	PD	OV	Z	AC	C
–	–	–	–	–	–	–	–

CLR WDT Clear watch dog timer

Description The WDT and the WDT Prescaler are cleared (re-counting from zero). The power down bit (PD) and time-out bit (TO) are cleared.

Operation WDT & WDT Prescaler ← 00H
PD & TO ← 0

Affected flag(s)

TC2	TC1	TO	PD	OV	Z	AC	C
–	–	0	0	–	–	–	–

CLR WDT1 Preclear watch dog timer

Description The PD, TO flags, WDT and the WDT Prescaler are cleared (re-counting from zero), if the other preclear WDT instruction had been executed. Only execution of this instruction without the other preclear instruction just sets the indicating flag which implies this instruction was executed and the PD and TO flags remain unchanged.

Operation WDT & WDT Prescaler ← 00H*
PD & TO ← 0*

Affected flag(s)

TC2	TC1	TO	PD	OV	Z	AC	C
–	–	0*	0*	–	–	–	–

CLR WDT2 Preclear watch dog timer

Description The PD, TO flags, WDT and the WDT Prescaler are cleared (re-counting from zero), if the other preclear WDT instruction had been executed. Only execution of this instruction without the other preclear instruction, sets the indicating flag which implies this instruction was executed and the PD and TO flags remain unchanged.

Operation WDT & WDT Prescaler ← 00H*
PD & TO ← 0*

Affected flag(s)

TC2	TC1	TO	PD	OV	Z	AC	C
–	–	0*	0*	–	–	–	–

CPL [m] Complement data memory

Description Each bit of the specified data memory is logically complemented (1's complement). Bits which previously contain a one are changed to zero and vice-versa.

Operation [m] ← $[\overline{m}]$

Affected flag(s)

TC2	TC1	TO	PD	OV	Z	AC	C
–	–	–	–	–	√	–	–

CPLA [m] Complement data memory–place result in accumulator

Description Each bit of the specified data memory is logically complemented (1's complement). Bits which previously contained a one are changed to zero and vice-versa. The complemented result is stored in the accumulator and the contents of the data memory remains unchanged.

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

Affected flag(s)

TC2	TC1	TO	PD	OV	Z	AC	C
–	–	–	–	–	√	–	–

DAA [m] Decimal-Adjust accumulator for addition

Description The accumulator value is adjusted to the BCD (Binary Code Decimal) code. The accumulator is divided into two nibbles. Each nibble is adjusted to the BCD code and an internal carry (AC1) will be done if the low nibble of the accumulator is greater than 9. The BCD adjustment is done by adding 6 to the original value if the original value is greater than 9 or a carry (AC or C) is set; otherwise the original value remains unchanged. The result is stored in the data memory and only the carry flag (C) may be affected.

Operation If ACC.3~ACC.0 > 9 or AC=1
then [m].3~[m].0 ← (ACC.3~ACC.0)+6, AC1=$\overline{AC}$
else [m].3~[m].0 ← (ACC.3~ACC.0), AC1=0
and
If ACC.7~ACC.4+AC1 > 9 or C=1
then [m].7~[m].4 ← ACC.7~ACC.4+6+AC1, C=1
else [m].7~[m].4 ← ACC.7~ACC.4+AC1, C=C

Affected flag(s)

TC2	TC1	TO	PD	OV	Z	AC	C
–	–	–	–	–	–	–	√

DEC [m] Decrement data memory

Description Data in the specified data memory is decremented by one.

Operation [m] ← [m]–1

Affected flag(s)

TC2	TC1	TO	PD	OV	Z	AC	C
–	–	–	–	–	√	–	–

DECA [m] Decrement data memory–place result in accumulator

Description Data in the specified data memory is decremented by one, leaving the result in the accumulator. The contents of the data memory remain unchanged.

Operation ACC ← [m]–1

Affected flag(s)

TC2	TC1	TO	PD	OV	Z	AC	C
–	–	–	–	–	√	–	–

HALT Enter power down mode

Description This instruction stops program execution and turn off the system clock. The contents of the RAM and registers are retained. The WDT and prescaler are cleared. The power down bit (PD) is set and the WDT time-out bit (TO) is cleared.

Operation PC ← PC+1
PD ← 1
TO ← 0

Affected flag(s)

TC2	TC1	TO	PD	OV	Z	AC	C
–	–	0	1	–	–	–	–

INC [m] Increment data memory

Description Data in the specified data memory is incremented by one.

Operation [m] ← [m]+1

Affected flag(s)

TC2	TC1	TO	PD	OV	Z	AC	C
–	–	–	–	–	√	–	–

INCA [m] Increment data memory–place result in accumulator

Description Data in the specified data memory is incremented by one, leaving the result in the accumulator. The contents of the data memory remain unchanged.

Operation

ACC ← [m]+1

Affected flag(s)

TC2	TC1	TO	PD	OV	Z	AC	C
–	–	–	–	–	√	–	–

JMP addr Direct Jump

Description Bits 0~10 of the program counter are replaced with the directly–specified address unconditionally, and control passed to this destination.

Operation PC ← addr

Affected flag(s)

TC2	TC1	TO	PD	OV	Z	AC	C
–	–	–	–	–	–	–	–

MOV A,[m] Move the contents of data memory to accumulator

Description The contents of the specified data memory is copied to the accumulator.

Operation ACC ← [m]

Affected flag(s)

TC2	TC1	TO	PD	OV	Z	AC	C
–	–	–	–	–	**	–	–

MOV A,x Move immediate data to accumulator

Description The 8–bit data specified by the code is loaded into the accumulator.

Operation ACC ← x

Affected flag(s)

TC2	TC1	TO	PD	OV	Z	AC	C
–	–	–	–	–	–	–	–

MOV [m],A Move contents of accumulator to data memory

Description The contents of the accumulator is copied to the specified data memory (one of the data memory).

Operation [m] ← ACC

Affected flag(s)

TC2	TC1	TO	PD	OV	Z	AC	C
–	–	–	–	–	–	–	–

NOP No operation

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

Operation PC ← PC+1

Affected flag(s)

TC2	TC1	TO	PD	OV	Z	AC	C
–	–	–	–	–	–	–	–

OR A,[m] Logical OR accumulator with data memory

Description Data in the accumulator and the specified data memory (one of the data memory) performs a bitwise logical_OR operation. The result is stored in the accumulator.

Operation ACC ← ACC "OR" [m]

Affected flag(s)

TC2	TC1	TO	PD	OV	Z	AC	C
–	–	–	–	–	√	–	–

OR A,x Logical OR immediate data to accumulator

Description Data in the accumulator and the specified data performs a bitwise logical_OR operation. The result is stored in the accumulator.

Operation ACC ← ACC "OR" x

Affected flag(s)

TC2	TC1	TO	PD	OV	Z	AC	C
–	–	–	–	–	√	–	–

ORM A,[m] Logical OR data memory with accumulator

Description Data in the data memory (one of the data memory) and the accumulator performs a bitwise logical_OR operation. The result is stored in the data memory.

Operation [m] ← ACC "OR" [m]

Affected flag(s)

TC2	TC1	TO	PD	OV	Z	AC	C
–	–	–	–	–	√	–	–

RET Return from subroutine

Description The program counter is restored from the stack. This is a two cycle instruction.

Operation PC ← Stack

Affected flag(s)

TC2	TC1	TO	PD	OV	Z	AC	C
–	–	–	–	–	–	–	–

RET A,x Return and place immediate data in accumulator

Description The program counter is restored from the stack and the accumulator loaded with the specified 8-bit immediate data.

Operation PC ← Stack
ACC ← x

Affected flag(s)

TC2	TC1	TO	PD	OV	Z	AC	C
–	–	–	–	–	–	–	–

RETI Return from interrupt

Description The program counter is restored from the stack, and interrupts enabled by setting the EMI bit. EMI is the enable master(global) interrupt bit (bit 0; register INTC).

Operation PC ← Stack
EMI ← 1

Affected flag(s)

TC2	TC1	TO	PD	OV	Z	AC	C
–	–	–	–	–	–	–	–

RL [m] Rotate data memory left

Description The contents of the specified data memory is rotated left one bit with bit 7 rotated into bit 0.

Operation [m].(i+1) ← [m].i; [m].i:bit i of the data memory (i=0-6)
[m].0 ← [m].7

Affected flag(s)

TC2	TC1	TO	PD	OV	Z	AC	C
–	–	–	–	–	–	–	–

RLA [m] Rotate data memory left–place result in accumulator

Description Data in the specified data memory is rotated left one bit with bit 7 rotated into bit 0, leaving the rotated result in the accumulator. The contents of the data memory remain unchanged.

Operation

ACC.(i+1) ← [m].i; [m].i:bit i of the data memory (i=0-6)
ACC.0 ← [m].7

Affected flag(s)

TC2	TC1	TO	PD	OV	Z	AC	C
–	–	–	–	–	–	–	–

RLC [m] Rotate data memory left through carry

Description The contents of the specified data memory and the carry flag are together rotated left one bit. Bit 7 replaces the carry bit; the original carry flag is rotated into the bit 0 position.

Operation [m].(i+1) ← [m].i; [m].i:bit i of the data memory (i=0-6)
[m].0 ← C
C ← [m].7

Affected flag(s)

TC2	TC1	TO	PD	OV	Z	AC	C
–	–	–	–	–	–	–	√

RLCA [m] Rotate left through carry–place result in accumulator

Description Data in the specified data memory and the carry flag are together rotated left one bit. Bit 7 replaces the carry bit and the original carry flag is rotated into bit 0 position. The rotated result is stored in the accumulator but the contents of the data memory remain unchanged.

Operation ACC.(i+1) ← [m].i; [m].i:bit i of the data memory (i=0-6)
ACC.0 ← C
C ← [m].7

Affected flag(s)

TC2	TC1	TO	PD	OV	Z	AC	C
–	–	–	–	–	–	–	√

RR [m] Rotate data memory right

Description The contents of the specified data memory are rotated right one bit with bit 0 rotated to bit 7.

Operation [m].i ← [m].(i+1); [m].i:bit i of the data memory (i=0-6)
[m].7 ← [m].0

Affected flag(s)

TC2	TC1	TO	PD	OV	Z	AC	C
–	–	–	–	–	–	–	–

RRA [m] Rotate right–place result in accumulator

Description Data in the specified data memory is rotated right one bit with bit 0 rotated into bit 7, leaving the rotated result in the accumulator. The contents of the data memory remain unchanged.

Operation ACC.(i) ← [m].(i+1); [m].i:bit i of the data memory (i=0-6)
ACC.7 ← [m].0

Affected flag(s)

TC2	TC1	TO	PD	OV	Z	AC	C
–	–	–	–	–	–	–	–

RRC [m] Rotate data memory right through carry

Description The contents of the specified data memory and the carry flag are together rotated right one bit. Bit 0 replaces the carry bit; the original carry flag is rotated into the bit 7 position.

Operation [m].i ← [m].(i+1); [m].i:bit i of the data memory (i=0-6)
[m].7 ← C
C ← [m].0

Affected flag(s)

TC2	TC1	TO	PD	OV	Z	AC	C
–	–	–	–	–	–	–	√

RRCA [m] Rotate right through carry–place result in accumulator

Description Data of the specified data memory and the carry flag are together rotated right one bit. Bit 0 replaces the carry bit and the original carry flag is rotated into the bit 7 position. The rotated result is stored in the accumulator. The contents of the data memory remain unchanged.

Operation ACC.i ← [m].(i+1); [m].i:bit i of the data memory (i=0-6)
ACC.7 ← C
C ← [m].0

Affected flag(s)

TC2	TC1	TO	PD	OV	Z	AC	C
–	–	–	–	–	–	–	√

SBC A,[m] Subtract data memory and carry from accumulator

Description The contents of the specified data memory and the complement of the carry flag are together subtracted from the accumulator, leaving the result in the accumulator.

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

Affected flag(s)

TC2	TC1	TO	PD	OV	Z	AC	C
–	–	–	–	√	√	√	√

SBCM A,[m] Subtract data memory and carry from accumulator

Description The contents of the specified data memory and the complement of the carry flag are together subtracted from the accumulator, leaving the result in the data memory.

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

Affected flag(s)

TC2	TC1	TO	PD	OV	Z	AC	C
–	–	–	–	√	√	√	√

SDZ [m] Skip if decrement data memory is zero

Description The contents of the specified data memory are decreased by one. If the result is zero, the next instruction is skipped. If the result is zero, the following instruction, fetched during the current instruction execution, is discarded and a dummy cycle replaced to get the proper instruction. This makes a 2 cycle instruction. Otherwise proceed with the next instruction.

Operation Skip if ([m]–1)=0, [m] ← ([m]–1)

Affected flag(s)

TC2	TC1	TO	PD	OV	Z	AC	C
–	–	–	–	–	–	–	–

SDZA [m] Decrement data memory–place result in ACC, skip if zero

Description The contents of the specified data memory are decreased by one. If the result is zero, the next instruction is skipped. The result is stored in the accumulator but the data memory remains unchanged. If the result is zero ,the following instruction, fetched during the current instruction execution, is discarded and a dummy cycle is replaced to get the proper instruction, that makes a 2 cycle instruction. Otherwise proceed with the next instruction.

Operation Skip if ([m]–1)=0, ACC ← ([m]–1)

Affected flag(s)

TC2	TC1	TO	PD	OV	Z	AC	C
–	–	–	–	–	–	–	–

SET [m] Set data memory

Description Each bit of the specified data memory is set to one.

Operation [m] ← FFH

Affected flag(s)

TC2	TC1	TO	PD	OV	Z	AC	C
–	–	–	–	–	–	–	–

SET [m].i Set bit of data memory

Description Bit "i" of the specified data memory is set to one.

Operation [m].i ← 1

Affected flag(s)

TC2	TC1	TO	PD	OV	Z	AC	C
–	–	–	–	–	–	–	–

SIZ [m] Skip if increment data memory is zero

Description The contents of the specified data memory is increased by one. If the result is zero, the following instruction, fetched during the current instruction execution, is discarded and a dummy cycle is replaced to get the proper instruction. This is a 2 cycle instruction. Otherwise proceed with the next instruction.

Operation Skip if ([m]+1)=0, [m] ← ([m]+1)

Affected flag(s)

TC2	TC1	TO	PD	OV	Z	AC	C
–	–	–	–	–	–	–	–

SIZA [m] Increment data memory–place result in ACC, skip if zero

Description The contents of the specified data memory is increased by one. If the result is zero, the next instruction is skipped and the result stored in the accumulator. The data memory remains unchanged. If the result is zero, the following instruction, fetched during the current instruction execution, is discarded and a dummy cycle replaced to get the proper instruction. This is a 2 cycle instruction. Otherwise proceed with the next instruction.

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

Affected flag(s)

TC2	TC1	TO	PD	OV	Z	AC	C
–	–	–	–	–	–	–	–

SNZ [m].i Skip if bit "i" of the data memory is not zero

Description If bit "i" of the specified data memory is not zero, the next instruction is skipped. If bit "i" of the data memory is not zero, the following instruction, fetched during the current instruction execution, is discarded and a dummy cycle is replaced to get the proper instruction. This is a 2 cycle instruction. Otherwise proceed with the next instruction.

Operation Skip if [m].i≠0

Affected flag(s)

TC2	TC1	TO	PD	OV	Z	AC	C
–	–	–	–	–	–	–	–

SUB A,[m] Subtract data memory from accumulator

Description The specified data memory is subtracted from the contents of the accumulator, leaving the result in the accumulator.

Operation ACC ← ACC+$\overline{[m]}$+1

Affected flag(s)

TC2	TC1	TO	PD	OV	Z	AC	C
–	–	–	–	√	√	√	√

SUBM A,[m] Subtract data memory from accumulator

Description The specified data memory is subtracted from the contents of the accumulator, leaving the result in the data memory.

Operation [m] ← ACC+$\overline{[m]}$+1

Affected flag(s)

TC2	TC1	TO	PD	OV	Z	AC	C
–	–	–	–	√	√	√	√

SUB A,x Subtract immediate data from accumulator

Description The immediate data specified by the code is subtracted from the contents of the accumulator, leaving the result in the accumulator.

Operation ACC ← ACC+$\overline{x}$+1

Affected flag(s)

TC2	TC1	TO	PD	OV	Z	AC	C
–	–	–	–	√	√	√	√

SWAP [m] Swap nibbles within the data memory

Description The low-order and high-order nibbles of the specified data memory (one of the data memory) are interchanged.

Operation [m].3~[m].0 ↔ [m].7~[m].4

Affected flag(s)

TC2	TC1	TO	PD	OV	Z	AC	C
–	–	–	–	–	–	–	–

SWAPA [m] Swap data memory–place result in accumulator

description The low-order and high-order nibbles of the specified data memory are interchanged, writing the result to 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)

TC2	TC1	TO	PD	OV	Z	AC	C
–	–	–	–	–	–	–	–

SZ [m] Skip if data memory is zero

Description If the contents of the specified data memory is zero, the following instruction, fetched during the current instruction execution, is discarded and a dummy cycle is replaced to get the proper instruction. This is a 2 cycle instruction. Otherwise proceed with the next instruction.

Operation Skip if [m]=0

Affected flag(s)

TC2	TC1	TO	PD	OV	Z	AC	C
–	–	–	–	–	–	–	–

SZA [m] Move data memory to ACC, skip if zero

Description The contents of the specified data memory is copied to accumulator. If the contents is zero, the following instruction, fetched during the current instruction execution, is discarded and a dummy cycle is replaced to get the proper instruction. This is a 2 cycle instruction. Otherwise proceed with the next instruction.

Operation Skip if [m]=0

Affected flag(s)

TC2	TC1	TO	PD	OV	Z	AC	C
–	–	–	–	–	–	–	–

SZ [m].i Skip if bit "i" of the data memory is zero

Description If bit "i" of the specified data memory is zero, the following instruction, fetched during the current instruction execution, is discarded and a dummy cycle is replaced to get the proper instruction. This is a 2 cycle instruction. Otherwise proceed with the next instruction.

Operation Skip if [m].i=0

Affected flag(s)

TC2	TC1	TO	PD	OV	Z	AC	C
–	–	–	–	–	–	–	–

TABRDC [m] Move ROM code (current page) to TBLH & data memory

Description The low byte of ROM code (current page) addressed by the table pointer (TBLP) is moved to the specified data memory and the high byte transferred to TBLH directly.

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

Affected flag(s)

TC2	TC1	TO	PD	OV	Z	AC	C
–	–	–	–	–	–	–	–

TABRDL [m] Move ROM code (last page) to TBLH & data memory

Description The low byte of ROM code (last page) addressed by the table pointer (TBLP) is moved to the data memory and the high byte transferred to TBLH directly.

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

Affected flag(s)

TC2	TC1	TO	PD	OV	Z	AC	C
–	–	–	–	–	–	–	–

XOR A,[m] Logical XOR accumulator with data memory

Description Data in the accumulator and the indicated data memory performs a bitwise logical Exclusive_OR operation and the result is stored in the accumulator.

Operation ACC ← ACC "XOR" [m]

Affected flag(s)

TC2	TC1	TO	PD	OV	Z	AC	C
–	–	–	–	–	√	–	–

XORM A,[m] Logical XOR data memory with accumulator

Description Data in the indicated data memory and the accumulator perform a bitwise logical Exclusive_OR operation. The result is stored in the data memory. The zero flag is affected.

Operation [m] ← ACC "XOR" [m]

Affected flag(s)

TC2	TC1	TO	PD	OV	Z	AC	C
–	–	–	–	–	√	–	–

XOR A,x Logical XOR immediate data to accumulator

Description Data in the the accumulator and the specified data perform a bitwise logical Exclusive_OR operation. The result is stored in the accumulator. The zero flag is affected.

Operation ACC ← ACC "XOR" x

Affected flag(s)

TC2	TC1	TO	PD	OV	Z	AC	C
–	–	–	–	–	√	–	–

Illustration of Application Circuit