HTVDS84

Magic Voice with Macro Instruction

Features

- Operating voltage: 2.4V~5.2V
- One CDS input pin
- Four input pins and eight I/O pins
- Two current type D/A switch outputs
- 2K×14 program ROM
- 80×8 data RAM
- Programmable volume
- Timebase interrupt
- Two sampling rate counters with overflow interrupts
- Watchdog Timer
- RC oscillator
- Up to 1μs instruction cycle
- 4MHz system clock
- HALT function and wake-up feature reduce power consumption
- 2-level subroutine nesting
- Bit manipulation instruction
- 14-bit table read instruction
- 63 powerful instructions
- All instructions in 1 or 2 machine cycles
- 8MHz system clock, V_{DD}=5V
- 28-pin SKDIP package

Applications

- Intelligent educational toys
- High end toy controllers
- Alert and warning systems
- Sound effect generators

General Description

The VDS84 series are 8-bit high performance microcontroller with voice synthesizer and tone generator. They are designed for applications on multiple I/Os with sound effects. The VDS84 series are excellent for versatile voice and sound effect product applications. They also include a HALT function to reduce power consumption.

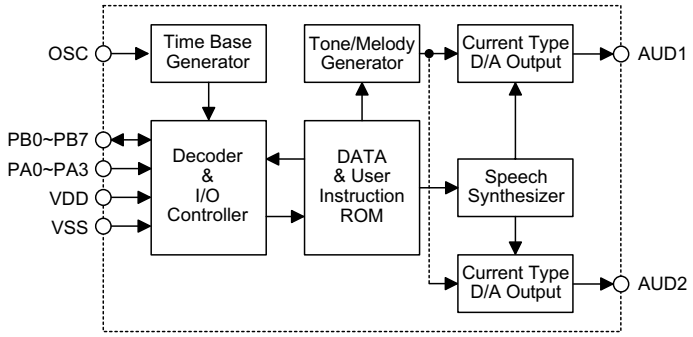
ROM Selection Table

The VDS84 series provides various voice capacity as shown below:

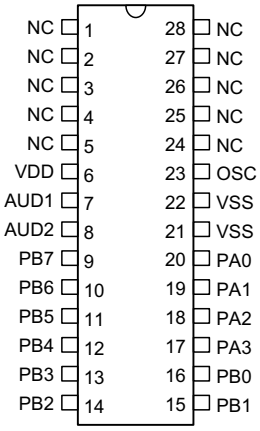
Part No.	HT84036	HT84072	HT84144	HT84192	HT84384
ROM	768Kb	1536Kb	3072Kb	4096Kb	8192Kb
Voice Length	36 sec	72 sec	144 sec	192 sec	384 sec

Note: The voice capacity is based on a sampling rate of 21Kb/s

Block Diagram



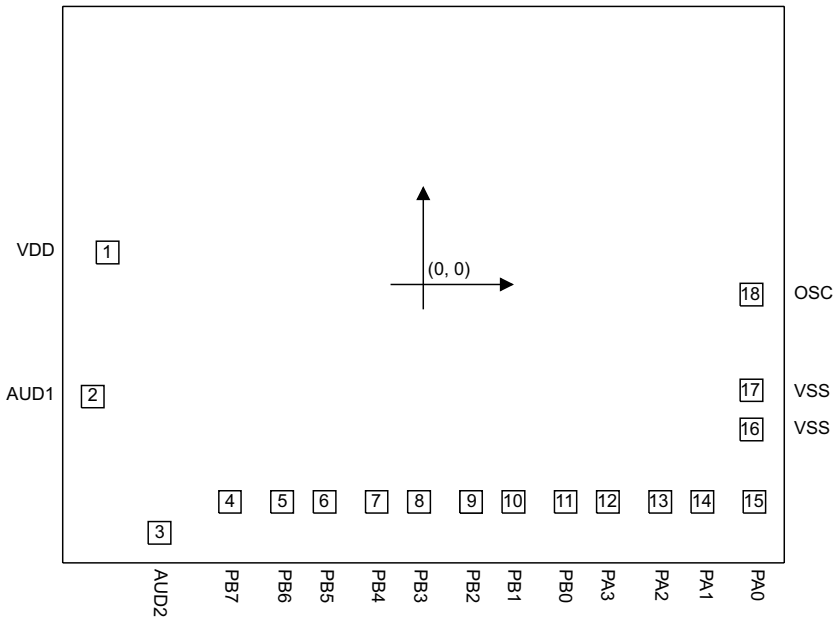
Pin Assignment



HT84XXX
– 28 SKDIP

Pad Assignment

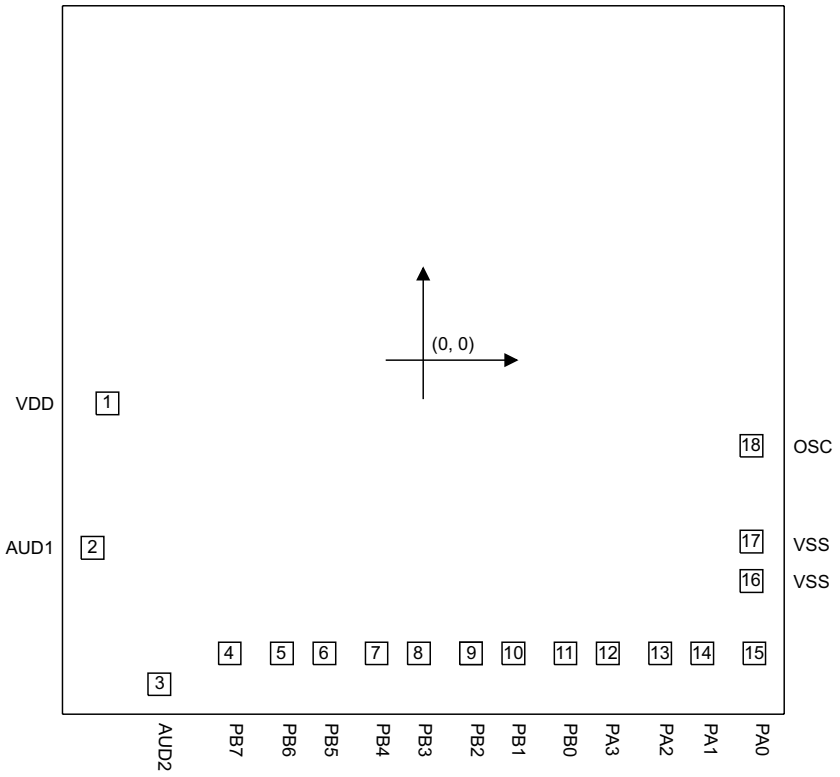
HT84036



Chip size: 2735 × 2105 $(\mu m)^2$

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

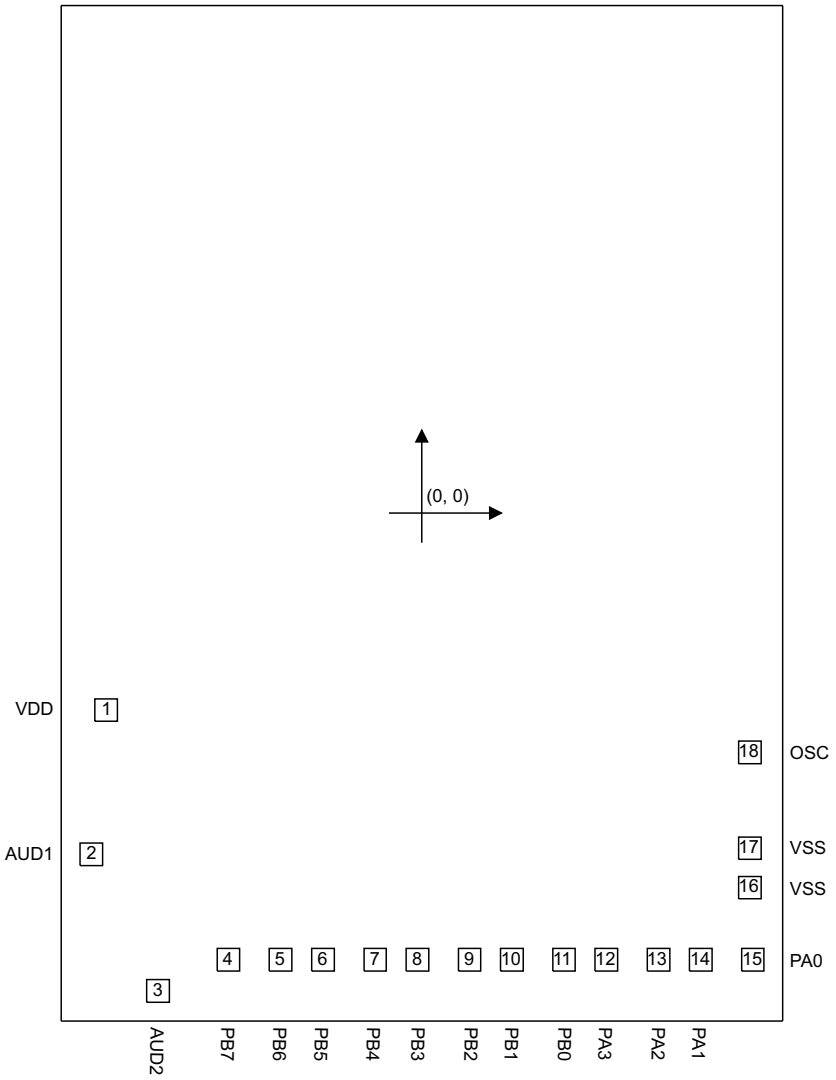
HT84072



Chip size: 2735 × 2645 $(\mu m)^2$

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

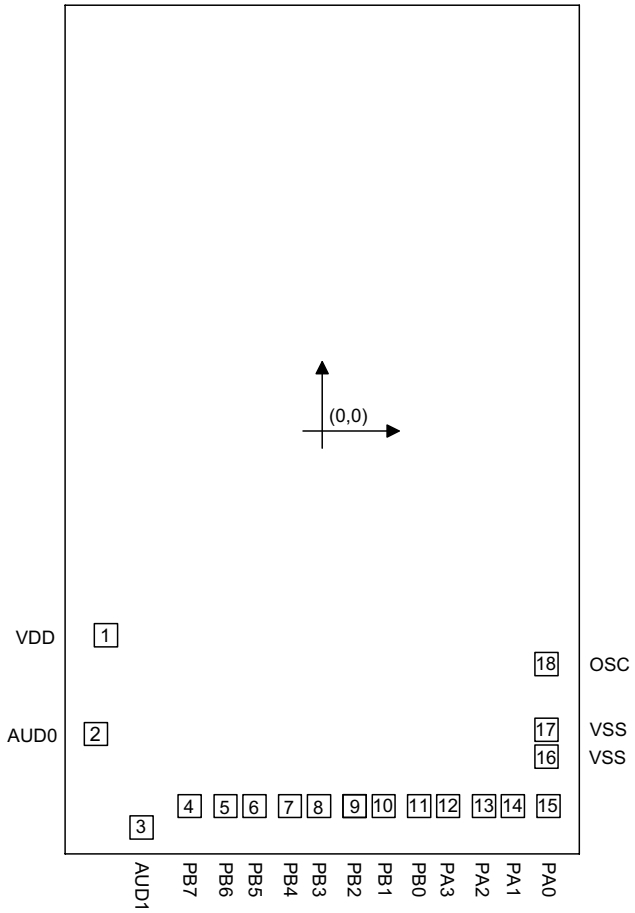
HT84144



Chip size: 2735 × 3735 $(\mu m)^2$

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

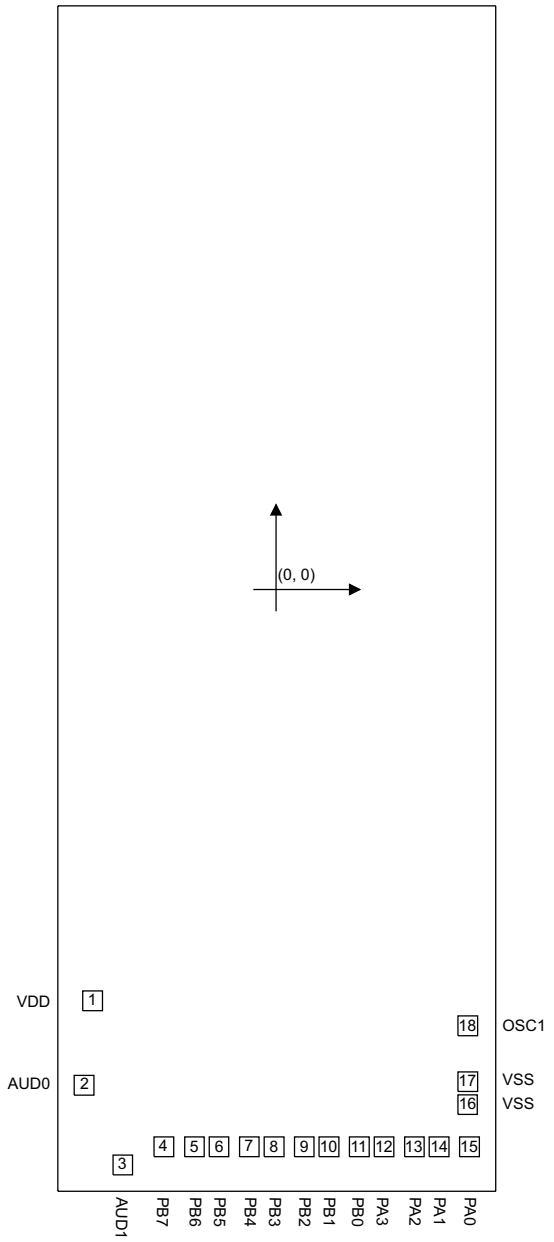
HT84192



Chip size: 2735 × 4460 (μm)^2

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

HT84384



Chip size: 2740 × 7350 $(\mu m)^2$

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

Pad Coordinates

HT84036 Unit: μm

Pad No.	X	Y	Pad No.	X	Y
1	–1125.31	114.74	10	320.15	–774.24
2	–1177.49	–398.79	11	505.75	–774.24
3	–939.68	–884.61	12	656.55	–774.24
4	–689.05	–774.24	13	842.15	–774.24
5	–503.45	–774.24	14	992.95	–774.24
6	–352.65	–774.24	15	1178.55	–774.24
7	–167.05	–774.24	16	1167.43	–517.18
8	–16.25	–774.24	17	1167.53	–377.08
9	169.35	–774.24	18	1167.53	–35.96

HT84072 Unit: μm

Pad No.	X	Y	Pad No.	X	Y
1	–1125.31	–155.26	10	320.15	–1044.24
2	–1177.49	–668.79	11	505.75	–1044.24
3	–939.68	–1154.61	12	656.55	–1044.24
4	–689.05	–1044.24	13	842.15	–1044.24
5	–503.45	–1044.24	14	992.95	–1044.24
6	–352.65	–1044.24	15	1178.55	–1044.24
7	–167.05	–1044.24	16	1167.43	–787.18
8	–16.25	–1044.24	17	1167.53	–647.08
9	169.35	–1044.24	18	1167.53	–305.96

HT84144 Unit: μm

Pad No.	X	Y	Pad No.	X	Y
1	–1125.31	–700.26	10	320.15	–1589.24
2	–1177.49	–1213.79	11	505.75	–1589.24
3	–939.68	–1699.61	12	656.55	–1589.24
4	–689.05	–1589.24	13	842.15	–1589.24
5	–503.45	–1589.24	14	992.95	–1589.24
6	–352.65	–1589.24	15	1178.55	–1589.24
7	–167.05	–1589.24	16	1167.43	–1332.18
8	–16.25	–1589.24	17	1167.53	–1192.08
9	169.35	–1589.24	18	1167.53	–850.96

HT84192 Unit: μm

Pad No.	X	Y	Pad No.	X	Y
1	–1125.31	–1062.76	10	320.15	–1951.74
2	–1177.49	–1576.29	11	505.75	–1951.74
3	–939.68	–2062.11	12	656.55	–1951.74
4	–689.05	–1951.74	13	842.15	–1951.74
5	–503.45	–1951.74	14	992.95	–1951.74
6	–352.65	–1951.74	15	1178.55	–1951.74
7	–167.05	–1951.74	16	1167.43	–1694.68
8	–16.25	–1951.74	17	1167.53	–1554.58
9	169.35	–1951.74	18	1167.53	–1213.46

HT84384 Unit: μm

Pad No.	X	Y	Pad No.	X	Y
1	–1121.81	–2507.96	10	323.65	–3396.94
2	–1173.99	–3021.49	11	509.25	–3396.94
3	–936.18	–3507.31	12	660.05	–3396.94
4	–685.55	–3396.94	13	845.65	–3396.94
5	–499.95	–3396.94	14	996.45	–3396.94
6	–349.15	–3396.94	15	1182.05	–3396.94
7	–163.55	–3396.94	16	1170.93	–3139.88
8	–12.75	–3396.94	17	1171.03	–2999.78
9	172.85	–3396.94	18	1171.03	–2658.66

Pin Description

Pin Name	I/O	Internal Connection	Description
PA0~PA3	I	Wake-up Pull-high	Trigger inputs Can also be configured as wake-up inputs
PB1~PB7	I/O	Pull-high or CMOS	Bidirectional I/O pins Can be optioned as trigger inputs or LED outputs
PB0	I/O	Pull-high or CMOS	Bidirectional I/O pins Can be mask optioned as CDS interface with internal Schmitt trigger input
AUD1	O	PMOS Open Drain	Audio output for driving an external transistor
AUD2	O	PMOS Open Drain	Audio output for driving an external transistor
OSC	I	—	Built-in RC oscillator An oscillator resistor is connected between OSC and VSS
VDD	—	—	Positive power supply
VSS	—	—	Negative power supply, ground

Absolute Maximum Ratings

Supply Voltage.....–0.3V to 6V

Input Voltage.....V_{SS}–0.3V to V_{DD}+0.3V

Storage Temperature.....–50°C to 125°C

Operating Temperature .....–20°C to 70°C

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.

Electrical Characteristics

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
V_{DD}	Operating Voltage	—	—	2.4	—	5.2	V
I_{STB}	Standby Current	3V	No load, system HALT	—	1	3	μA
I_{DD}	Operating Current	3V	No load, f_{SYS}=4MHz	—	5	8	mA
I_{OL}	PB (0:7) Sink Current	3V	V_{OL}=0.3V	4	6	—	mA
I_O	Max. AUD1 and AUD2 Output Current	3V	V_{OH}=0.6V	–1.5	–2	—	mA
f_{SYS}	System Frequency	3V	R_{OSC}=180kΩ	3.6	4.0	4.4	MHz

Functional Description

Executive flow

The system clock for the VDS84 series is derived from an RC type of oscillator. The clock is internally divided into four non-overlapping clocks denoted by P1, P2, P3 and P4. An instruction cycle consists of T1~T4.

Instruction fetching and execution are pipelined in such a way that a fetch takes an instruction cycle while decoding and execution take the next instruction cycle. The pipelining scheme causes each instruction to execute effectively in a cycle. If an instruction changes the program counter, two cycles are required to complete that instruction.

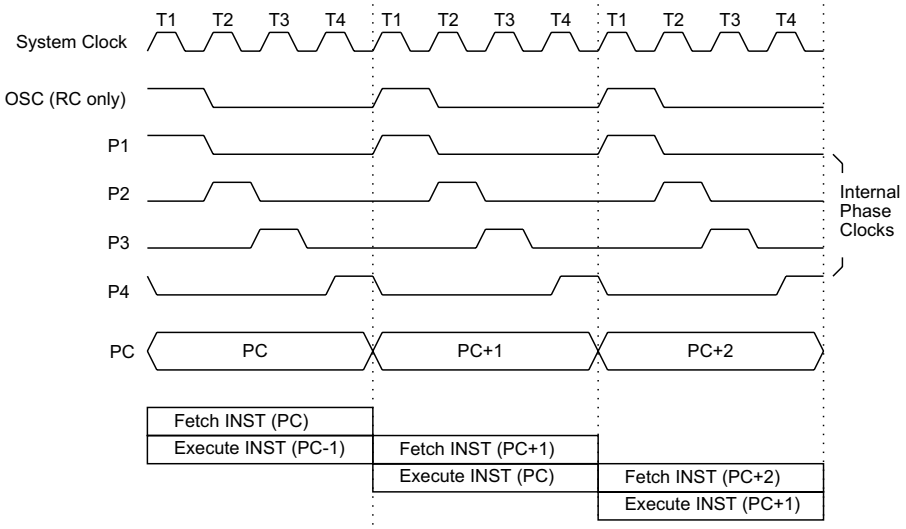
Program counter – PC

The 11-bit program counter (PC) controls the sequence in which the instructions stored in the program ROM are executed and the ROM space are 2048×14 bits. The contents of the program counter are incremented by one after a program memory word is accessed to fetch an instruction code. The program counter then points to a memory word containing the next instruction code.

The PC manipulates a program transfer by loading the address corresponding to each instruction when executing a jump instruction, conditional skip execution, loading PCL register, subroutine call, initial reset, internal interrupt or return from subroutine.

The conditional skip is activated by instructions. Once the condition is satisfied, the next instruction, fetched during the current instruction execution, is discarded and a dummy cycle replaces it to get a proper instruction. Otherwise, the system will proceed with the next instructions.

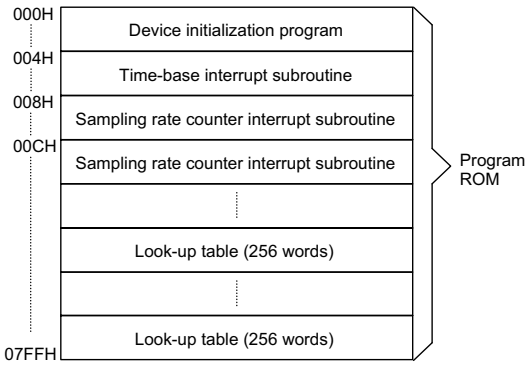
Once a control transfer takes place, the execution requires an additional dummy cycle.



Executive flow

Program memory – ROM

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 for VDS84 is 2048×14. Certain locations in the program memory are reserved for special usage:



- Location 000H

  This area is reserved for program initialization. The program always begins execution at location 000H each time the system is reset.

- Location 004H

  This area is reserved for a Timebase interrupt service program. If the Timebase counter overflows, the interrupt is enabled and the stack is not full, the program begins execution at location 004H.

- Location 008H

  This area is reserved for voice sampling rate counter 1 interrupt service program. If the timer interrupt results from a sampling rate counter overflow, the interrupt is enabled and the stack is not full, the program begins execution at location 008H.

- Location 00CH

  This area is reserved for voice sampling rate counter 2 interrupt service program. If the timer interrupt results from a sampling rate counter overflow, the interrupt is enabled and the stack is not full, the program begins execution at location 00CH.

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
Timebase Overflow	0	0	0	0	0	0	0	0	1	0	0
Sampling Rate Counter 1 Overflow	0	0	0	0	0	0	0	1	0	0	0
Sampling Rate Counter 2 Overflow	0	0	0	0	0	0	0	1	1	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

Program counter

Note: *10~*0: Program counter bits
#10~#0: Instruction code bits
S10~S0: Stack register bits
@7~@0: PCL bits

Stack register – Stack

The stack register is a special part of the memory used to save the contents of the program counter (PC). This stack is organized into two levels. It is neither part of the data nor part of the program space, and cannot be read or written to. Its activated level is indexed by a stack pointer (SP) and cannot be read or written to. At a subroutine call or interrupt acknowledgment, the contents of the program counter are pushed onto the stack. The program counter is restored to its previous value from the stack at the end of a subroutine or interrupt routine, which is signaled by a return instruction (RET or RETI). After a chip resets, SP will point to the top of the stack.

The interrupt request flag will be recorded but the acknowledgment will be inhibited when the stack is full and a non-masked interrupt takes place. After the stack pointer is decremented (by RET or RETI), the interrupt will be serviced. This feature prevents stack overflow and allows programmers to use the structure more easily. In a similar case, if the stack is full and a "CALL" is subsequently executed, stack overflow occurs and the first entry is lost.

Data memory – RAM

The data memory is designed with 103x 8 bits. The data memory is further divided into two functional groups; namely, special function registers and general purpose data memory (80x8). Although most of them can be read or be written to, some are read only.

The special function registers include an indirect addressing register (R0; 00H), memory pointer register (MP0; 01H), accumulator (ACC; 05H), program counter lower-order byte register (PCL; 06H), table pointer (TBLP; 07H), table higher-order byte register (TBLH; 08H), status register (STATUS; 0AH), interrupt control register (INTC; 0BH), I/O registers (PA; 12H, PB; 14H) and I/O control registers (PAC; 13H, PBC; 15H), clock MUX register of sampling rate register (TMRC; 1DH). The 20H to 2DH are used for sound and tone (melody) synthesis. The function registers include a lower-order byte register (PDAL; 20H, 24H) of

00H	R0	*R/W
01H	MP0	*R/W
02H		
03H		
04H		
05H	ACC	*R/W
06H	PCL	*R/W
07H	TBLP	*R/W
08H	TBLH	*R
09H		
0AH	STATUS	*R/W
0BH	INTC	*R/W
0CH ... 11H		
12H	PA	*R/W
13H	PAC	*R/W
14H	PB	*R/W
15H	PBC	*R/W
16H ... 1CH		
1DH	TMRC	*R/W
1EH		
1FH		
20H	PDA1L	*R/W
21H	PDA1H	*R/W
22H	VOL1	*R/W
23H	SR1	*R/W
24H	PDA2L	*R/W
25H	PDA2H	*R/W
26H	VOL2	*R/W
27H	SR2	*R/W
28H ... 2BH		
2CH	CROM	*R/W
2DH	CROMC	*W
2EH		
2FH		
30H ... 7FH	General Purpose Data Memory	*R/W

Note: *R/W: readable/writable

*R: read only

*W: write only

D/A data, higher-order byte register (PDAH; 21H, 25H) of D/A data, volume control register (VOL; 22H, 26H), sampling rate control register (SR; 23H, 27H), voice ROM data register (CROM; 2CH) and control register (CROMC; 2DH).

The general-purpose data memory (30H - 7FH) is used for data and control formation under instruction commands.

All of the areas of data memory can handle arithmetic, logic, increment, decrement and rotate operations directly. Except for some dedicated bits, each bit in the data memory can be set and reset by "SET [m]." and "CLR [m].i", and can also be indirectly accessed through a memory pointer register (MP0; 01H).

Indirect addressing register

Location 00H is an indirect addressing register that is not physically implemented. Any read/write operation of R0 (00H) accesses the data memory pointed to by MP0 (01H). Indirectly reading location 00H will return the result to 00H whereas; indirectly writing it will have no effect. The bit 7 of MP0 is undefined. Reading it will return the result "1" and any writing operation to MP0 will transfer only the lower 7-bit data to MP0.

Accumulator

The accumulator is closely related to ALU operations. It is also mapped to location 05H of the data memory and is capable of carrying out immediate data operations. The data movement between two data memory locations must pass 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 and decrement (INC, DEC)
- Branch decision (SZ, SNZ, SIZ, SDZ ....)

The ALU not only saves the results of a data operation but can also change the status register.

Program counter low-order byte register – PCL

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

Table pointer and table high-order byte – TBLP & TBLH

Any location in the program 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 higher-order byte register (TBLH) of the table is read only. The table pointer (TBLP) is a read/write register (07H) which indicates the table location. This location must be placed in TBLP before accessing the table. The TBLH is read only and cannot be restored.

Instruction	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

Table location

Note: *10~*0: Table location bits P10~P6: Current PC value bits

7@~ @0 : Table pointer bits

If the main routine and the ISR (Interrupt Service Routine) both employ the table read instruction, the contents of the TBLH in the main routine are likely to be changed by the table read instruction used in the ISR. Errors can occur. In other words, using the table read instruction in the main routine and the ISR simultaneously should be avoided. However, if the table read instruction has to be applied in both the main routine and the ISR, the interrupt is supposed to be disabled prior to the table read instruction. It will not be enabled until the TBLH has been backed up. All table related instructions require two cycles to complete the operation. These areas may function as normal program memory depending upon the requirements.

Status register – STATUS

This 8-bit register (0AH) consists of a zero flag (Z), carry flag (C), auxiliary carry flag (AC), overflow flag (OV), power down flag (PD) , watchdog time-out flag (TO) and timebase counter indicator flag (TF). It also records the status information and controls the operation sequence.

Except for the TO and PD flags, bits in the status register can be altered by instructions similar to other registers. Any data written into the status register will not change the TO or PD flag. Operations related to the status register may yield different results from those intended. The TO and PD flags can be altered only by a Watchdog Timer overflow, chip power-up, clearing the watchdog time or executing the "HALT" instruction. TF flag is toggle output 1 or 0, and be transferred by timebase counter overflow .

The Z, OV, AC and C flags generally reflect the status of the latest operations. The status register will not be automatically pushed onto the stack on entering the interrupt sequence or executing the subroutine call. If the status contents are important and if the subroutine may corrupt the status register, the programmer must take precautions and save it properly.

Interrupt

The VDS84 provide a Timebase interrupt in addition to two sampling rate counter interrupts. The interrupt control register (INTC; 0BH) includes interrupt control bits to set the enable/disable and the interrupt request flags.

Labels	Bits	Function
C	0	C is set if the 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. It is also affected by a rotate through carry instruction.
AC	1	AC is set if the 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 takes place; otherwise AC is cleared.
Z	2	Z is set if the result of an arithmetic or logic operation is zero; otherwise Z is cleared.
OV	3	OV is set if the 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.
PD	4	PD is cleared by a system power-up or executing the "CLR WDT" instruction. PD is set by executing the "HALT" instruction.
TO	5	TO is cleared by a system power-up or executing the "CLR WDT" or "HALT" instructions. TO is set by a WDT time-out.
—	6	Undefined, read as "0"
TF	7	TF flag is toggle output 1 or 0, and be transferred by timebase Counter overflow.

Status register – STATUS

Once an interrupt subroutine is serviced, all the other interrupts will be blocked (by clearing the EMI bit). This scheme may prevent any further interrupt nesting. Other interrupt requests may happen during this interval but only the interrupt request flag is recorded. If an interrupt needs 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 the SP is decremented, whether or not the related interrupt is enabled. If immediate service is desired, the stack has to be prevented from becoming full.

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

Timebase interrupt is triggered by a 3-stage counter overflow. The related interrupt request flag (F1MS; bit 4 of INTC) are set . When related control bits are enabled (EMI,ETBI=1 and STE=0) , the stack is not full and the Timebase interrupt is active, a subroutine call to location 04H will occur. The interrupt request flag (F1MS) and EMI bits will be cleared automatically to disable other interrupts.

The sampling rate counter interrupt is initialized by setting a sampling rate counter interrupt request flag (FS1/FS2; bit 5/6 of INTC), which is caused by the counter overflow. When the related control bits are enabled (EMI,SR1I/SR2I=1), the stack is not full and the sampling rate interrupt is active, a subroutine call to location 08H/0CH will occur. The related interrupt request flag (FS1/FS2) and the EMI bit will be cleared automatically 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 the related interrupt control bit are set to 1 (if the stack is not full). To return from an interrupt subroutine, "RET" or "RETI" may be invoked. RETI will set the EMI bit to enable an interrupt service, but RET will not.

Interrupts occurring in an interval between the rising edges of two consecutive T2 pulses will be serviced at the latter of the two T2 pulses if the corresponding interrupts are enabled. In the case of simultaneous requests, they can be masked by resetting the EMI bit. The Priority table illustrates the priority of applying the simultaneous requests:

Register	Bit No.	Label	Function
INTC (0BH)	0	EMI	Controls the master (global) interrupt (1: enable; 0: disable)
	1	ETBI	Controls the Timebase interrupt (solve) (1: enable; 0: disable)
	2	ES1I	Controls the Sampling rate counter 1 interrupt (1: enable; 0: disable)
	3	ES2I	Controls the Sampling rate counter 2 interrupt (1: enable; 0: disable)
	4	F1MS	Timebase counter request flag (1: active; 0: inactive)
	5	FS1	Sampling rate counter 1 request flag (1: active; 0: inactive)
	6	FS2	Sampling rate counter 2 request flag (1: active; 0: inactive)
	7	STE	Controls the Timebase interrupt (master) (1: disable; 0: enable)

INTC register

No.	Interrupt Source	Priority	Vector
a	Time-base counter	1	04H
b	Sampling rate 1 counter	2	08H
c	Sampling rate 2 counter	3	0CH

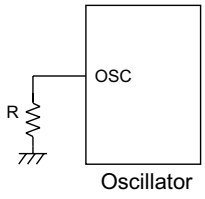
TABLE of the interrupt priority

The sampling rate 1 counter interrupt request flag (FS2), sampling rate 0 counter interrupt request flag (FS1), Timebase interrupt request flag (F1MS), enable sampling rate 1 counter bit (SR2I), enable sampling rate 0 counter bit (SR1I), enable Timebase bit (ETBI,STE) and enable master interrupt bit (EMI) make up an interrupt control register (INTC) which is located at 0BH in the data memory. EMI, ETBI, SBMI, SR1I and SR2I are used to control the enable/disable status of interrupts. These bits prevent the requested interrupt from being serviced. Once the interrupt request flags (F1MS, FS1, FS2) are all set, they will remain in the INTC register till the interrupts are serviced or cleared by a software instruction.

The "CALL subroutine" is preferably not used within the interrupt subroutine. This is because interrupts often occur in an unpredictable manner or required to be serviced immediately in certain applications. If only one stack is left and enabling the interrupt is not well controlled, operation of the "call" in the interrupt subroutine will damage the original control sequence.

Oscillator configuration

The VDS84 provides RC oscillator circuits, for system clocks. When the device enters the HALT mode, the system oscillator stops to conserve power. An external resistor between OSC and GND is required and the range of the resistance has to be from 91K to 180K. The RC type of oscillator provides the most cost-effective solution. Nonetheless, 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 required.



Watchdog Timer – WDT

The WDT clock source is implemented by an instruction clock (system clock divided by 4). The Watchdog Timer is designed to prevent software malfunction or sequence jumping to an unknown location with unpredictable results. It can be disabled by mask option. After it is disabled, all executions related to WDT are ignored. WDT is first divided by 4096 (12 stages) to get a nominal time-out period of 1ms (under Fsys=4MHz) once an internal WDT oscillator is selected. Then the WDT register is divided by 7, so the WDT time-out period is 7ms (under Fsys=4MHz). (See the Fig. Sampling counter and Timebase and IRQ Diagram)

The WDT overflow under a normal operation initializes a "chip reset" and sets the status bit "TO" It will initialize a "warm reset", and only PC and SP are reset to zero. To clear the contents of the WDT register, two methods are adopted, namely, software instructions, and "HALT" instruction. The software instructions include "CLR WDT" and the other sets - "CLR WDT1" and "CLR WDT2". Of these two types of instructions, by mask option only one can be active at a time . If "CLR WDT" is chosen (i.e., "one instruction" selected), any execution of the "CLR WDT" instruction will clear the WDT.

In the case that "CLR WDT1" and "CLR WDT2" are selected (i.e., "two instruction" selected) , these two instructions must be executed to clear the WDT; otherwise WDT may reset the chip as a result of time-out.

The state of the register is summarized in the following table:

Register	Power On	WDT Time-out	WDT Time-out from HALT
PC	0000H	0000H	0000H
MP0	-xxx xxxx	-uuu uuuu	-uuu uuuu
ACC	xxxx xxxx	uuuu uuuu	uuuu uuuu
TBLP	xxxx xxxx	uuuu uuuu	uuuu uuuu
TBLH	--xx xxxx	--uu uuuu	--uu uuuu
STATUS	x-00 xxxx	x-1u uuuu	x-11 uuuu
INTC	0000 0000	0000 0000	uuuu uuuu
PA	----1111	---- 1111	---- uuuu
PAC	----1111	---- 1111	---- uuuu
PB	1111 1111	1111 1111	uuuu uuuu
PBC	1111 1111	1111 1111	uuuu uuuu
TMRC	-xxx -xxx	-uuu -uuu	-uuu -uuu
SR1/2	xxxx xxxx	uuuu uuuu	uuuu uuuu
PDAL1/2	0000----	0000 ----	0000 ----
PDAH1/2	0000 0000	0000 0000	0000 0000
VOL1/2	0000----	0000 ----	0000 ----
CROM	xxxx xxxx	uuuu uuuu	uuuu uuuu
CROMC	---- --xx	---- --uu	---- --uu

Note: "u" means unchanged "x" means unknown

Power down operation – HALT

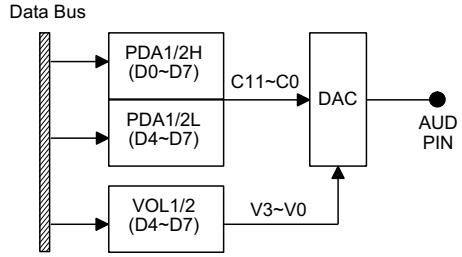
The HALT mode is initialized by the "HALT" instruction and results in the following:

- The system oscillator is turned off
- The contents of the on-chip RAM and registers remain unchanged
- The WDT register is cleared
- All the I/O ports maintain their original status
- The PD flag is set and the TO flag cleared
- To turn off the DAs, the values are 0.

The system can quit the HALT mode by external falling edge signal on port A or port B. Each bit in port A or port B can be independently selected to wake up the device by mask option. After the TO and PD flags are examined, the reason for chip reset can be determined. The PD flag is cleared by system power-up or executing the CLR WDT instruction and is set when executing the HALT instruction . The TO flag is set if the WDT time-out occurs, and causes a wake-up that only resets the PC and SP; the others maintain their original status.

The port A or port B wake-up can be considered as a continuation of normal execution. Each bit in port A or port B can be independently selected to wake up the device by the mask option. Awakening from an I/O port stimulus, the program will resume execution of the next instruction. Once a wake-up event occurs, it takes 1024 tsys (system clock period) to resume normal operation. In other words, a dummy period will be inserted after wake-up. To minimize power consumption, all the I/O pins should be carefully managed before entering the HALT mode.

Audio output and volume control



Audio output diagram

The VDS84 series provide two 12-bits current type DAC devices for driving external 8 ohm speaker through an external NPN transistor. The programmer must write the voice data to the register PDAL (20H/24H) and PDAH (21H/25H). Only 12 bits which include the high nibble of PDAL and the whole byte of PDAH are used. The correct procedure for DAC output as below , high nibble data of PDAL must be written at first, and then the PDAH data is written second.

There are 16 scales of volume controllable level that are provided for the current type DAC output. The programmer only writes the volume control data to the VOL register (22H/26H). The VOL1 (22H) control PDA1H-PDA1L and VOL2 (26H) control PDA2H-PDA2L. Only the high nibble of VOL are used. Note that writing 0H to the high nibble of VOL doesn't denote mute output , this mean still leakage from AUD pin through external NPN transistor , and external 8 ohm speaker leak current too. Only load 00h to PDAH-PDAL will turn off DAC and prevent leakage

Sampling rate counter – SR1, SR2, TMRC

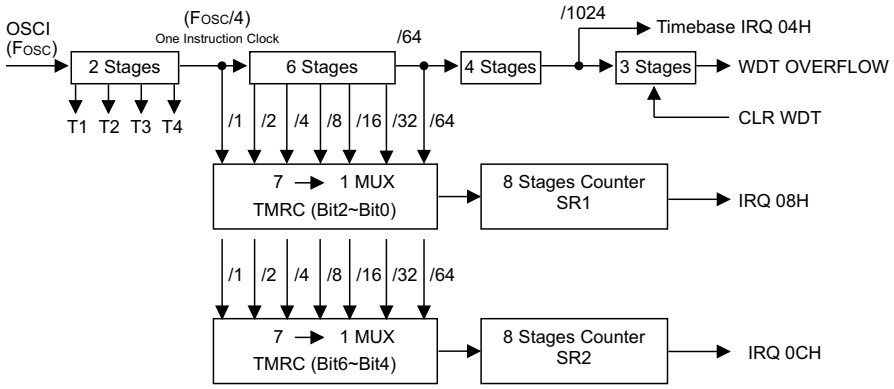
The VDS84 series offer two sampling rate counter SR1(23H) and SR2 (27H). Those counters contain an 8-bit programmable count-up counter. The sampling rate counter clock source is decided by the TMRC (1DH) control register. The TMRC defines the sampling rate counter clock source as shown in the table below:

TMRC (1DH)				SR0 Clock Source
Bit3	Bit2	Bit1	Bit0	
—	0	0	0	Disable counter
—	0	0	1	1 instruction clock /1
—	0	1	0	1 instruction clock /2
—	0	1	1	1 instruction clock /4
—	1	0	0	1 instruction clock /8
—	1	0	1	1 instruction clock /16
—	1	1	0	1 instruction clock /32
—	1	1	1	1 instruction clock /64
TMRC (1DH)				SR1 Clock Source
Bit7	Bit6	Bit5	Bit4	
—	0	0	0	Disable counter
—	0	0	1	1 instruction clock /1
—	0	1	0	1 instruction clock /2
—	0	1	1	1 instruction clock /4
—	1	0	0	1 instruction clock /8
—	1	0	1	1 instruction clock /16
—	1	1	0	1 instruction clock /32
—	1	1	1	1 instruction clock /64

Note: One instruction clock equate one system clock divided by 4
TMRC bit3 and bit7 are unused

Two physical registers are mapped to SR1 location; writing SR1 makes the starting value be placed in the counter preload register and reading SR1/2 gets the contents of the counter.

When the counter (reading SR1/2) is read, the clock will be blocked to avoid errors. As clock blocking may result in a counting error, this must be taken into consideration by the programmer.



Sampling counter & Timebase & IRQ diagram

Voice ROM

The VDS84 series embed an 8-bit width voice ROM for storing sound and tone (melody) data. Coded data can be saved in an internal mask ROM by changing one layer of the mask after the sound and tone (melody) sources are coded by Holtek's tools. The voice ROM size for VDS84 series are shown bellow:

Body	Voice ROM (Byte)	P-ROM (Word)	RAM (Byte)	PA	PB	DAC (Sets)	Voice ROM Address Counter
HT84036	96K	2K	80	4-bit	8-bit	2	20-bit×2
HT84072	192K	2K	80	4-bit	8-bit	2	20-bit×2
HT84144	384K	2K	80	4-bit	8-bit	2	20-bit×2
HT84192	512K	2K	80	4-bit	8-bit	2	20-bit×2
HT84384	1024K	2K	80	4-bit	8-bit	2	20-bit×2

The handshaking between the microcontroller and voice ROM is through a ROM data register (CROM. 2CH) and control register (CROMC; 2DH).

The voice ROM with two 20-bit address counter decoder, writing address to the address counter which are selected by bit1 of CROMC (0: counter 1 , 1: counter 2). Load the contents of the address counter to the voice ROM cell for reading, and that which contains the address counter are selected by bit0 of CROMC (0: counter 1 , 1: counter 2). The bit 7 to bit 2 of CROMC are not valid.

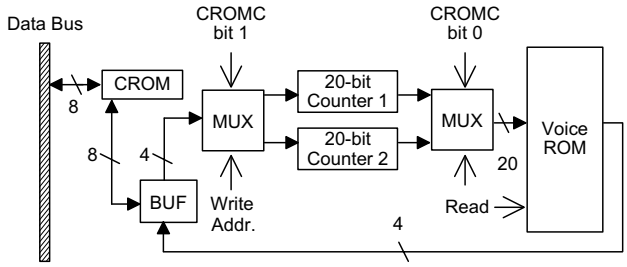
The related voice ROM address has to be written in the address counter first if the microcontroller attempts to read the sound or tone (melody) data in the voice ROM. After the microcontroller finishes reading a byte of data, its internal address counter will automatically be incremented by one. In this case, reading continuous data only requires loading the starting address to the CROM .

The lower-order nibble is valid whereas the higher-order nibble is not valid in the CROM . Based on this difference, the start address has to be divided into six nibbles, and written into the CROM six times. The lower-order nibble and higher-order nibble data can then be read back by reading twice after the sound or tone (melody) starting address is written.

The reading of one byte of data must delay at least twenty instruction cycles after the address is exchanged however the address is programmed or automatically increase by one. For example, if the starting address of the sound data to be read is 0CF0H by address counter 1, the program of reading one byte of sound or tone (melody) data is as shown below:

```
; Selected #1 address counter
; Set reflashes to #1
; Set read to #1
MOV     A, 0011b
MOV     CROMC, A
; Write the first nibble address
MOV     A, 00H
MOV     CROM, A
; Write the second nibble address
MOV     A, 0FH
MOV     CROM, A
; Write the third nibble address
MOV     A, 0CH
MOV     CROM, A
; Write the fourth nibble address
MOV     A, 00H
MOV     CROM, A
; Write the fifth and sixth addresses
CLR     CROM
CLR     CROM
; Delay 20 instruction cycles
CALL    DELAY

; Read the lower order nibble data
SWAPA   CROM
; Read the high-order nibble data
OR      A, CROM
SWAP    ACC
```

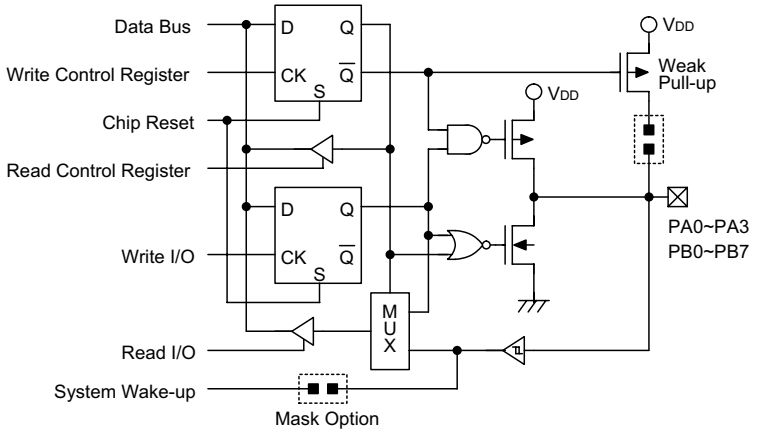


Voice ROM

Input/output ports

The VDS84 series include four input line and eight bidirectional input/output lines, labeled PA and PB which are mapped to the data memories of [12H], [14H], respectively. PA is input only, PB can be used as input and output operations. For input operation, these ports are non-latched, i.e., the inputs must be ready at the T2 rising edge of the instruction "MOV A, [m]" (m=12H, 14H). For output operation, all the data are latched and remain unchanged until the output latch is rewritten.

PB has its own control register (PBC) to control the input/output configuration. With a control register, a CMOS output or schmitt trigger input can be reconfigured dynamically (i.e., on the fly) under a software control. To function as an input, the corresponding latch of a control register must write "1". The pull-high resistance will be automatically exhibited . The input source also depends on the control register. If the bit of the control register bit is "1", the input will read the pad state. If it is "0", the contents of the latches will move to the internal bus. The latter is possible only in the "read-modify-write" instruction. For the output function, CMOS is the only configuration. These control registers are all mapped to locations [15H]. PA is input only and functions the same as PB.



Input/output ports

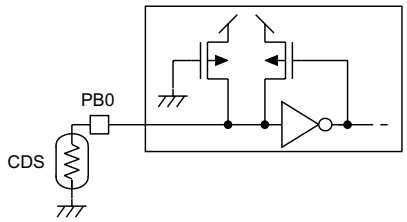
These input/output lines remain at a high level after a chip reset. Each bit of the input/output latches can be set or cleared by the "SET [m].i" and "CLR [m].i" (m=12H, 14H) instructions.

Some instructions will first input data and then follow the output operations. For instance, "SET [m].i", "CLR [m].i", "CPL[m]", and "CPLA[m]" read the entire port state into the CPU, execute the defined operations (bit-operation), and then write the results back to the latches or accumulator. Each line of port A and port B is capable of waking up the device. The highest four bits of port A are not physically implemented. A "0" will return to reading the highest four bits, but writing them will result with no operation.

CDS interface pin

The PB0 pin be allowed as CDS input pin by mask option.

An active pull-high resistor constructs a schmitt trigger structure and improves the efficiency of input detection. The pull-high resistor is about 900kΩ when the input voltage of PB0 is lower than V_{IL}. It turns out to be about 200kΩ when the input voltage is higher than V_{IH}.



Mask option

The following table illustrates four kinds of mask option in the VDS84 series. All of them have to be defined to ensure a proper functioning system.

No.	Mask Option
1	WDT enable or disable
2	CLRWDT times selection . This option defines how to clear WDT by instruction . One time means that the CLR WDT instruction can clear the WDT . Two times means that if both of the CLR WDT1 and CLR WDT2 instruction have been executed, only then the WDT can be cleared.
3	Wake-up selection. This option defines the wake up capability. External I/O pins (PA and PB) all have the capability to wake up the chip from a HALT.
4	Pull-high value selection. This option decides either 33k or 98k ohm resistor is active in the input mode. Each bit of the I/O port can be independently selected.

MACRO Directives

• Macro command

Macro Command	Description
Speech	Play the sound files indicated by the TrackNumber
SamplingRate	Assign sampling rate for SR1 or SR2 interrupt
SetAddress	Set address
ReadByte	Read a byte from the counter ROM to accumulator
FuncSetAddress	Set address by calling procedure
FuncReadByte	Read a byte from the counter ROM to accumulator by calling procedure

• Symbol and variable

Symbol and Variable	Description
STEREO	Define two channel output (AUD1 and AUD2)
TrackNumber	Record the sound file order (0~255)
Tempo	Record the melody tempo
TimeBaseCnt	Use as a down counter

• Flag

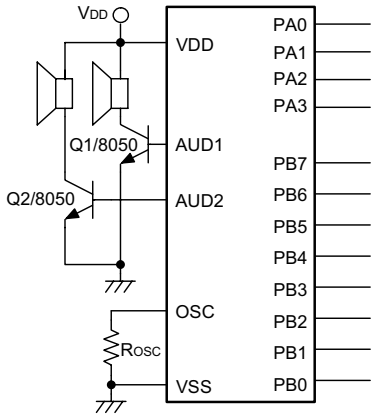
Flag	Description
FStandBy	FStandBy is set if the playing sound is terminative
FPause	FPause is set for pausing the playing sound
FStop	FStop is set for stopping the playing sound
FNoteChanged	FNoteChanged is set if the note of the playing melody is changed
FVoiceDown	FVoiceDown will clear AUD output when the sound ends
FChannelNo	FChannelNo will assign the playing channel

• f_{OSC}—R_{OSC} table

f_{OSC}	R_{OSC}
4MHz±10%	180kΩ
5MHz±10%	150kΩ
6MHz±10%	120kΩ
8MHz±10%	91kΩ

V_{DD}=5V

Application Circuits



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, 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 and 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
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

Mnemonic	Description	Flag Affected
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 unconditionally	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
Miscellaneous		
NOP	No operation	None
CLR [m]	Clear data memory	None
SET [m]	Set data memory	None
CLR WDT	Clear the Watchdog Timer	TO,PD
CLR WDT1	Pre-clear the Watchdog Timer	TO*,PD*
CLR WDT2	Pre-clear the 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

Note: x: 8-bit immediate data
m: 8-bit data memory address
i: 0~7 number of bits
addr: 11-bit program memory address
A: accumulator
*: 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 the Watchdog 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 and WDT Prescaler ← 00H
PD and TO ← 0

Affected flag(s)

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

CLR WDT1 Preclear the Watchdog 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 that this instruction was executed and the PD and TO flags remain unchanged.

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

Affected flag(s)

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

CLR WDT2 Preclear the Watchdog Timer

Description The PD and 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 that this instruction was executed and the PD and TO flags remain unchanged.

Operation WDT and WDT Prescaler ← 00H*
PD and 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 $[m] \leftarrow [\overline{m}]$

Affected flag(s)

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

CPLA [m] Complement data memory and 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 remain unchanged.

Operation ACC ← [$\overline{m}$]

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 BCD code and an internal carry (AC1) will be generated 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
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 and 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 the program execution and turns 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 and 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 0~11 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 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 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 the interrupts are 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 and 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 and 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 and 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 and 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 decremented 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 and place result in ACC, skip if zero

Description The contents of the specified data memory are decremented 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 to 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 incremented 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 to 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 and place result in ACC, skip if zero

Description The contents of the specified data memory is incremented 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 to 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 to 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+$\bar{x}$+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 to 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 to 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 to 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 and 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 and 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
—	—	—	—	—	√	—	—

Holtek Semiconductor Inc. (Headquarters)
No.3, Creation Rd. II, Science-based Industrial Park, Hsinchu, Taiwan, R.O.C.
Tel: 886-3-563-1999
Fax: 886-3-563-1189

Holtek Semiconductor Inc. (Taipei Office)
11F, No.576, Sec.7 Chung Hsiao E. Rd., Taipei, Taiwan, R.O.C.
Tel: 886-2-2782-9635
Fax: 886-2-2782-9636
Fax: 886-2-2782-7128 (International sales hotline)

Holtek Semiconductor (Hong Kong) Ltd.
RM.711, Tower 2, Cheung Sha Wan Plaza, 833 Cheung Sha Wan Rd., Kowloon, Hong Kong
Tel: 852-2-745-8288
Fax: 852-2-742-8657

Holtek Semiconductor (Shanghai) Ltd.
7th Floor, Building 2, No.889, Yi Shan Rd., Shanghai, China
Tel: 021-6485-5560
Fax: 021-6485-0313

Holmate Technology Corp.
48531 Warm Springs Boulevard, Suite 413, Fremont, CA 94539
Tel: 510-252-9880
Fax: 510-252-9885