IS23SC4456

Contactless CPU Card Chip

USER MANUAL - PRELIMINARY

Table of Contents

1 Introduction

IS23SC4456 is targeting at contactless smart card applications such as e-transfer, e-ticket, electronic purse, security access and multi-applications in one-card purposes.

IS23SC4456 offers 8K bytes of STS-ROM, 24K bytes of User-ROM (for user COS debugging purpose, an emulation IC whose user-ROM is replaced with EEPROM can be provided), 256 bytes internal RAM, 768 bytes XRAM, 256 bytes RF-buffer and 8K bytes EEPROM, which can be used as both data and program memory. The non-volatile memory consists of high reliability cells to guarantee data integrity. This is especially important when the EEPROM is used as program memory.

IS23SC4456 has two versions: EVA and MASK: In MASK version, the 24K User-ROM is made of ROM, and the user COS is masked into the ROM before fab-out. In EVA version, the 24K User-ROM is made of EEPROM and the chip has In System Programming (ISP) feature, user can download their COS into the code memory using ISP, for detail information about ISP please refer to “IS23SC4456_AN001.pdf”.

IS23SC4456 RF interface conform to ISO/IEC 14443 Type A, and fully compliant with S50. The S50 feature can be turned off permanently.

2 Features

2.1 Basic

- 0.18u EEPROM technology
- Conform to ISO/IEC 14443 Type A, and fully compliant with S50
- 8-bit low power Turbo 8051 CPU
- ESD protection greater than 6KV (HBM)
- Support ISO14443-4 , and 106K/212Kbps transmission

2.2 Memory

- 8K bytes EEPROM
- 24K bytes User-ROM, 8K bytes STS ROM
- 256 bytes internal RAM, 1K bytes XRAM (768 bytes common XRAM + 256 bytes RF-buffer)
- Flexible EEPROM page mode from 1 to 32 byte
- Typical EEPROM program time < 5 ms @ 1.8V
- EEPROM data retention minimum 10 years
- EEPROM minimum program cycles: 100,000

2.3 Security feature

- MOVC block from 8K-EEPROM code
- Memory data/program encryption without performance penalty
- Address and data scramble
- Low voltage sensor
- Triple DES
- Cipher stream mechanism
- True random number generator

3 Pin assignment

Table 3-1 Pin Description

Pin Name	**Function**	**Special Note**
RF1	Coil connection pin RF1	
RF2	Coil connection pin RF2	

4 Block diagram

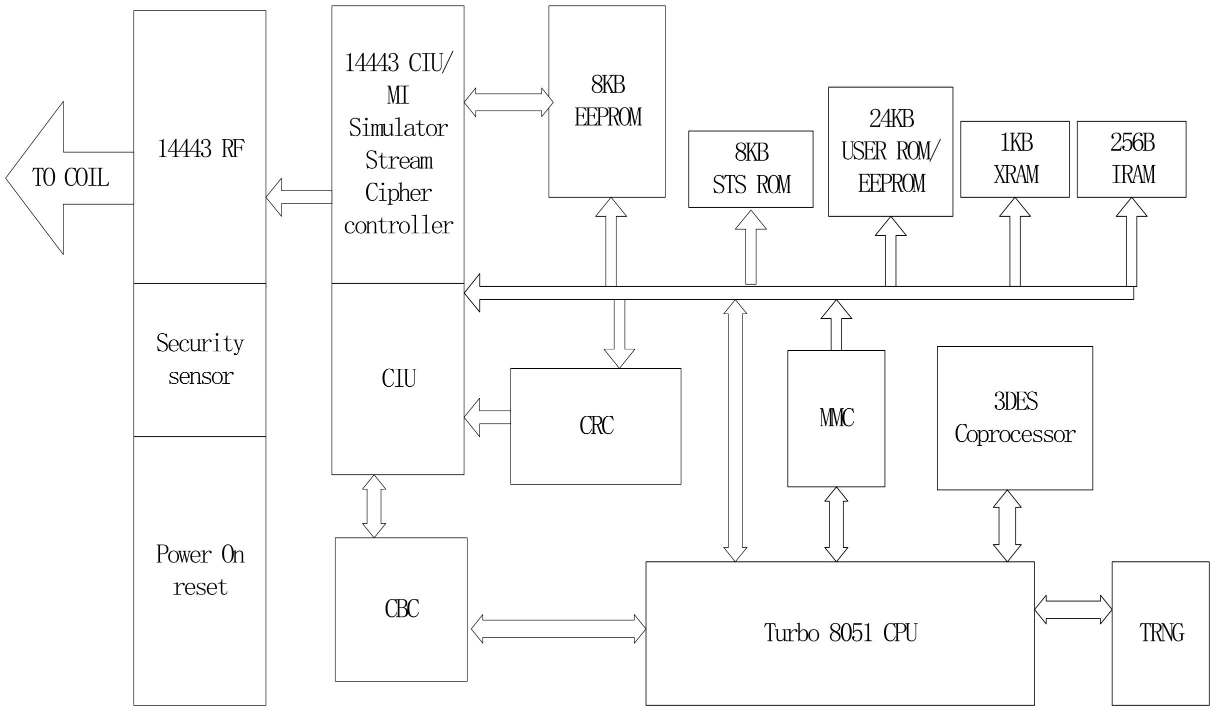

Figure 4-1 block diagram

5 CPU

5.1 Overview

IS23SC4456's CPU has extended memory addressing capabilities. It executes all the standard 8051 instructions approximately 3 times faster in terms of number of clock cycles.

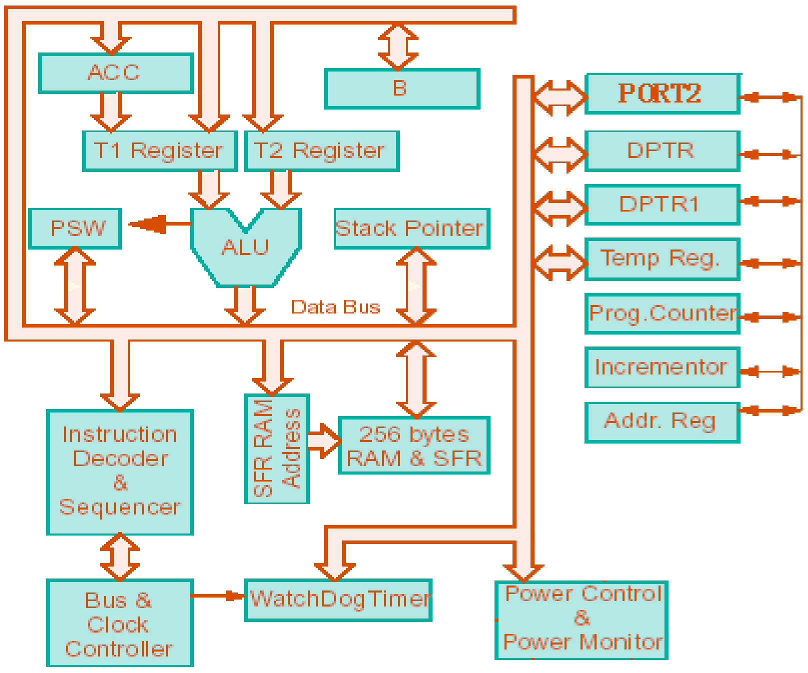

Figure 5-1 CPU Architecture

5.2 Register

5.2.1 B (F0H)

Table 5-1 B Register (B)

B	7	6	5	4	3	2	1	0
BIT	B.7	B.6	B.5	B.4	B.3	B.2	B.1	B.0
RST	0	0	0	0	0	0	0	0
W/R	W/R	W/R	W/R	W/R	W/R	W/R	W/R	W/R

This is an 8-bit register that is used as the second argument in the MUL and DIV instructions. For all other instructions it can be used simply as a general-purpose register.

5.2.2 ACC (E0H)

The Accumulator (ACC) is the primary register used in arithmetic, logical and data transfer operations in the CPU. Since the Accumulator is directly accessible by the CPU, most of the high-speed instructions make use of the ACC as one argument.

Table 5-2 ACCUMULATOR (ACC)

ACC	7	6	5	4	3	2	1	0

BIT	ACC.7	ACC.6	ACC.5	ACC.4	ACC.3	ACC.2	ACC.1	ACC.0
RST	0	0	0	0	0	0	0	0
W/R	W/R	W/R	W/R	W/R	W/R	W/R	W/R	W/R

5.2.3 PSW (D0H)

It is used to store the status bits of the ALU. It holds the Carry flag, the Auxiliary Carry flag, General-purpose flags, the Register Bank Select, the Overflow flag, and the Parity flag.

Table 5-3 Program Status Word (PSW)

PSW	7	6	5	4	3	2	1	0
BIT	CY	AC	RSV	RS1	RS0	OV	RSV	P
RST	0	0	0	0	0	0	0	0
W/R	W/R	W/R	W/R	W/R	W/R	W/R	W/R	W/R

- CY: Carry flag, set for an arithmetic operation, which results in a carry being generated from the ALU. It is also used as the accumulator for the bit operations.
- AC: Auxiliary carry, set when the previous operation resulted in a carry (during addition) or a borrow (during subtraction) from the high order nibble.
- RS.1-0, register bank select bits.

RS1	RS0	Register bank	Address
0	0	0	00-07H
0	1	1	08-0FH
1	0	2	10-17H
1	1	3	18-1FH

- OV: Overflow flag, set when a carry was generated from the seventh bit but not from the 8th bit as a result of the previous operation.
- P: Parity flag, set/cleared by hardware to indicate odd/even number of 1's in the acc.

5.2.4 Stack

The IRAM can be used for the stack. This area is selected by the Stack Pointer (SP), which stores the address of the top of the stack. Whenever a jump, call or interrupt is invoked the return address is placed on the stack. There is no restriction as to where the stack can begin in the IRAM. By default however, the Stack Pointer contains 07h (IRAM location) at reset. The user can then change this to any value desired. The SP will point to the last used value. Therefore, the SP will be incremented and then address is saved onto the stack. Conversely, while popping from the stack the contents will be read first, then the SP is decremented.

5.2.4.1 SP (81H)

Table 5-4 STACK POINTER (SP)

SP	7	6	5	4	3	2	1	0
BIT	SP.7	SP.6	SP.5	SP.4	SP.3	SP.2	SP.1	SP.0
RST	0	0	0	0	0	1	1	1
W/R	W/R	W/R	W/R	W/R	W/R	W/R	W/R	W/R

5.2.5 Data Pointer

The Data Pointers are used in MOVX instructions that transfer data to and from the External Data memories. These pointers hold the address of the External Memory location with which data is to be transferred. IS23SC4456 has an additional Data Pointer. Both the data pointers are again extendable by way of additional data pointer pair. The additional pointer holds upper bits of extended memory address.

5.2.5.1 DPL (82h)

Table 5-5 DATA POINTER LOW (DPL)

DPL	7	6	5	4	3	2	1	0
BIT	DPL.7	DPL.6	DPL.5	DPL.4	DPL.3	DPL.2	DPL.1	DPL.0
RST	0	0	0	0	0	0	0	0
W/R	W/R	W/R	W/R	W/R	W/R	W/R	W/R	W/R

This is the low byte of the 24-bit data pointer.

5.2.5.2 DPH (83H)

Table 5-6 DATA POINTER MIDDLE (DPH)

DPH	7	6	5	4	3	2	1	0
BIT	DPH.7	DPH.6	DPH.5	DPH.4	DPH.3	DPH.2	DPH.1	DPH.0
RST	0	0	0	0	0	0	0	0
W/R	W/R	W/R	W/R	W/R	W/R	W/R	W/R	W/R

This is the middle byte of the 24-bit data pointer.

5.2.5.3 DPX (93H)

Table 5-7 DATA POINTER HIGH (DPX)

DPX	7	6	5	4	3	2	1	0
BIT	DPX.7	DPX.6	DPX.5	DPX.4	DPX.3	DPX.2	DPX.1	DPX.0
RST	0	0	0	0	0	0	0	0
W/R	W/R	W/R	W/R	W/R	W/R	W/R	W/R	W/R

This is the most significant byte of the 24-bit data pointer.

5.2.5.4 DPL1 (84h)

Table 5-8 DATA POINTER LOW 1 (DPL1)

DPL1	7	6	5	4	3	2	1	0
BIT	DPL1.7	DPL1.6	DPL1.5	DPL1.4	DPL1.3	DPL1.2	DPL1.1	DPL1.0
RST	0	0	0	0	0	0	0	0
W/R	W/R	W/R	W/R	W/R	W/R	W/R	W/R	W/R

This is the low byte of the 24-bit data pointer

The user can switch between DPL, DPH, DPX and DPL1, DPH1, DPX1 simply by setting DPS.

If DPS =1, the instructions that use DPTR will now access DPL1, DPH1 and DPX1 in place of DPL, DPH and DPX. If they are not required the user can use them as conventional register locations.

5.2.5.5 DPH1 (85H)

Table 5-9 DATA POINTER MIDDLE 1 (DPH1)

DPH1	7	6	5	4	3	2	1	0
BIT	DPH1.7	DPH1.6	DPH1.5	DPH1.4	DPH1.3	DPH1.2	DPH1.1	DPH1.0
RST	0	0	0	0	0	0	0	0
W/R	W/R	W/R	W/R	W/R	W/R	W/R	W/R	W/R

This is the middle byte of 24-bit data pointer.

The user can switch between DPL, DPH, DPX and DPL1, DPH1, DPX1 simply by setting DPS.

If DPS =1, the instructions that use DPTR will now access DPL1, DPH1 and DPX1 in place of DPL, DPH and DPX. If they are not required the user can use them as conventional register locations.

5.2.5.6 DPX1 (95H)

Table 5-10 DATA POINTER HIGH (DPX1)

DPX1	7	6	5	4	3	2	1	0
BIT	DPX1.7	DPX1.6	DPX1.5	DPX1.4	DPX1.3	DPX1.2	DPX1.1	DPX1.0
RST	0	0	0	0	0	0	0	0
W/R	W/R	W/R	W/R	W/R	W/R	W/R	W/R	W/R

This is the most significant byte of the 24-bit data pointer.

The user can switch between DPL, DPH, DPX and DPL1, DPH1, DPX1 simply by setting DPS.

If DPS =1, the instructions that use DPTR will now access DPL1, DPH1 and DPX1 in place of DPL, DPH and DPX. If they are not required the user can use them as conventional register locations.

5.2.5.7 DPS (86H)

Table 5-11 DATA POINTER SELECT (DPS1)

DPS	7	6	5	4	3	2	1	0
BIT	RSV	RSV	RSV	RSV	RSV	RSV	RSV	DPS
RST	0	0	0	0	0	0	0	0
W/R	R	R	R	R	R	R	R	W/R

- DPS, DPS is used to select between the DPTR and DPTR1.

  0: Select DPTR as 24-bit data pointer

  1: Select DPTR1 as 24-bit data pointer

5.2.6 PORT

5.2.6.1 P2 (A0H)

Table 5-12 P2 Register (P2)

P2	7	6	5	4	3	2	1	0
BIT	P2.7	P2.6	P2.5	P2.4	P2.3	P2.2	P2.1	P2.0
RST	1	1	1	1	1	1	1	1
W/R	W/R	W/R	W/R	W/R	W/R	W/R	W/R	W/R

This register is concatenated with MXAX, R1 or R0 to form 24-bit address when executing MOVX A,@Ri or MOVX @Ri,A In this case P2[7:0] gives the middle 8bits of address.

5.2.6.2 MXAX (EAH)

Table 5-13 MOVX Extended Address Register (MXAX)

MXAX	7	6	5	4	3	2	1	0
BIT	MXAX.7	MXAX.6	MXAX.5	MXAX.4	MXAX.3	MXAX.2	MXAX.1	MXAX.0
RST	0	0	0	0	0	0	0	0
W/R	W/R	W/R	W/R	W/R	W/R	W/R	W/R	W/R

This register is concatenated with P2, R1 or R0 to form 24-bit address when executing MOVX A,@Ri or MOVX @Ri,A in 24-bit paged mode. In this case MXAX[7:0] gives the most significant bits of address.

5.3 Memory Organization

There are 3 kinds of memory within IS23SC4456: Internal Data Memory (IMEM), External Data Memory (XMEM) and Program Memory (PMEM).

The IMEM only includes 256 bytes scratch-pad RAM (IRAM). This can be used by the user for temporary storage during program execution. A certain section of this RAM is bit addressable, and can be directly addressed for this purpose.

The XMEM includes the 1K bytes SRAM (XRAM), 8K bytes EEPROM. They are used to store data for memory mapped devices.

The PMEM includes 24K bytes ROM/EEPROM and 8K bytes EEPROM. They are used to store the instruction OPCODE.

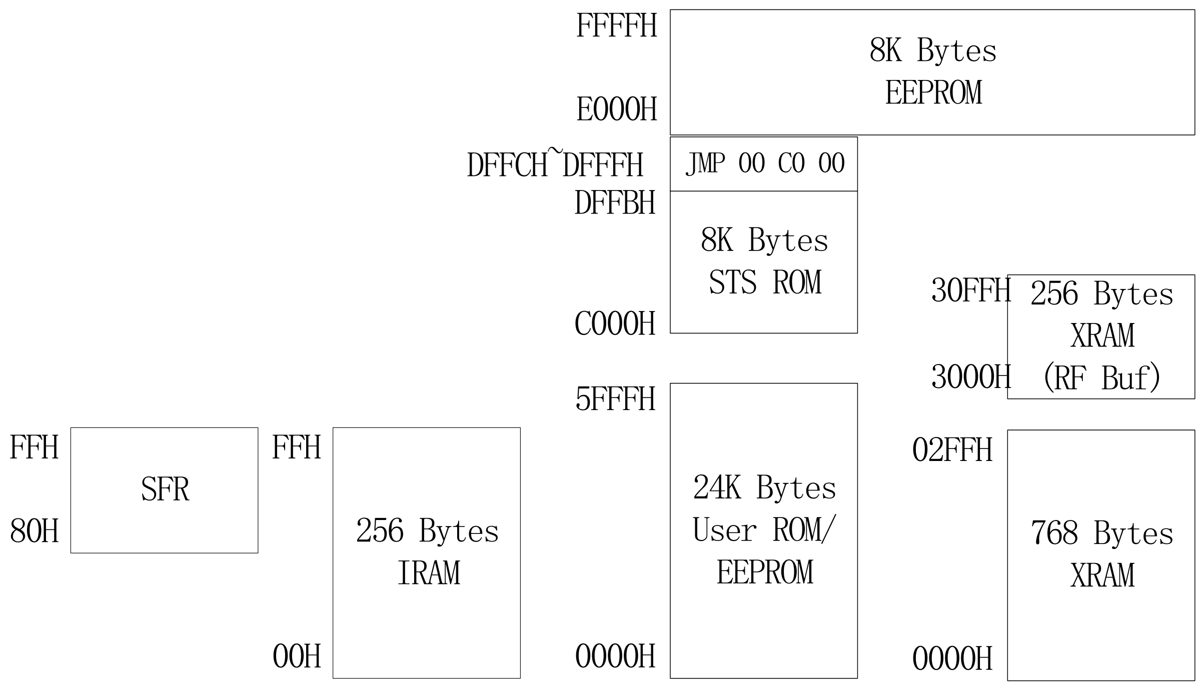

Figure 5-2 Memory organization

The code from DFFCH~DFFFH of the program space is a branch instruction "JMP ADDR" which branches the PC to STS entrance.

5.4 Register Map

There are several Special Function Registers (SFRs), which can be accessed by software only in direct addressing mode, while the IRAM can be accessed by either direct or indirect addressing. Usually IRAM is used when data contents are small. In case of larger data contents are present, XMEM will be used. However, the IRAM has the fastest access times. There are several other special purpose areas within the IRAM. These are described as follows.

5.4.1 Working Registers

There are four sets of working registers, each consisting of eight 8-bit registers. These are termed as Banks 0, 1, 2, and 3. Individual registers within these banks can be directly accessed by separate instructions. These individual registers are named as R0, R1, R2, R3, R4, R5, R6 and R7. However, at one time IS23SC4456 can work with only one particular bank. The bank selection is done by setting PSW.*RS1-RS0* bits. The R0 and R1 registers are used to store the address for indirect accessing.

5.4.2 Bit addressable Locations

The IRAM area from location 20h to 2Fh is byte as well as bit addressable. This means that a bit in this area can be individually addressed. In addition, some of the other SFRs are also bit addressable. In the SFR area, any existing SFR whose address ends in a 0 or 8 is bit addressable.

Table 5-14: IRAM address

FFH 80H	Indirect addressing internal RAM							
7FH 30H	Direct Addressing internal RAM							
2FH	7F	7E	7D	7C	7B	7A	79	78
2EH	77	76	75	74	73	72	71	70
2DH	6F	6E	6D	6C	6B	6A	69	68
2CH	67	66	65	64	63	62	61	60
2BH	5F	5E	5D	5C	5B	5A	59	58
2AH	57	56	55	54	53	52	51	50
29H	4F	4E	4D	4C	4B	4A	49	48
28H	47	46	45	44	43	42	41	40
27H	3F	3E	3D	3C	3B	3A	39	38
26H	37	36	35	34	33	32	31	30
25H	2F	2E	2D	2C	2B	2A	29	28
24H	27	26	25	24	23	22	21	20
23H	1F	1E	1D	1C	1B	1A	19	18
22H	17	16	15	14	13	12	11	10
21H	0F	0E	0D	0C	0B	0A	09	08
20H	07	06	05	04	03	02	01	00
1FH 18H	Bank 3							
17H 10H	Bank 2							
0FH 08H	Bank1							
07H 00H	Bank0							

5.4.3 Special Function Registers

IS23SC4456 uses Special Function Registers (SFRs) to control and monitor peripherals and their modes. The SFRs reside in the register locations 80-FFh and are accessed by direct addressing only. Some of the SFRs are bit addressable. This is very useful in cases where we wish to modify a particular bit without changing the others. The SFRs that are bit addressable are those whose addresses end in 0 or 8. The list of the SFRs is as follows.

Table 5-15 SFR LOCATION TABLE

F8	EIP							
F0	B							
E8	EIE		MXAX					

E0	ACC							
D8						DES_CON	DES_KEY	DES_DAT
D0	PSW							
C8							CRC_CON	CRC_DAT
C0							RNGD	TA
B8	IP							
B0								
A8	IE			BUF_CON	BUF_STATUS	BUF_SIZE	BIT_CNT	
A0	P2					BAUDSEL	SYS_CLK_CON	
98							SECCON	
90				DPX		DPX1		EE_PGM_CON
88								EPCON
80		SP	DPL	DPH	DPL1	DPH1	DPS	PCON

6 Interrupts

6.1 Overview

The IS23SC4456 has a two-priority level interrupt structure with 2 interrupt sources. Each of the interrupt sources has an individual priority bit, flag, interrupt vector and enable bit. In addition, the interrupts can be globally enabled or disabled .The interrupt sources are as follow:

- RF receiving
- EEPROM write finish

6.2 Register

6.2.1 IE (A8H)

Table 6-1 INTERRUPT ENABLE (IE)

IE	7	6	5	4	3	2	1	0
BIT	EA	RSV	RSV	RSV	RSV	EEW	RSV	RSV
RST	0	0	0	0	0	0	0	0
W/R	W/R	R	R	R	R	W/R	R	R

- EA : global interrupt enable

  0: Disable all interrupts

  1: Enable all interrupts.

- EEW

  0: Disable EEPROM write finish interrupt

  1: Enable EEPROM write finish interrupt

6.2.2 IP (B8H)

Table 6-2 INTERRUPT PRIORITY (IP)

IP	7	6	5	4	3	2	1	0
BIT	RSV	RSV	RSV	RSV	RSV	PEW	RSV	RSV
RST	1	0	0	0	0	0	0	0
W/R	R	R	R	R	R	W/R	R	R

- PEW: Priority of EEPROM write finish interrupt

  0: set EEPROM write finish interrupt priority to lower level.

  1: set EEPROM write finish interrupt priority to higher level

6.2.3 EIE (E8H)

Table 6-3 Extended Interrupt Enable (EIE)

EIE	7	6	5	4	3	2	1	0

BIT	RSV	RSV	RSV	RSV	RSV	RSV	RSV	ERFR
RST	0	0	0	0	0	0	0	0
W/R	R	R	R	R	R	R	R	W/R

- ERFR

  0: Disable RF receiving interrupt

  1: Enable RF receiving interrupt

6.2.4 EIP (F8H)

Table 6-4 Extended Interrupt Priority (EIP)

EIP	7	6	5	4	3	2	1	0
BIT	RSV	RSV	RSV	RSV	RSV	RSV	RSV	PRFR
RST	0	0	0	0	0	0	0	0
W/R	R	R	R	R	R	R	R	W/R

- PRFR: Priority of RF receiving interrupt

  0: Set RF receiving interrupt priority to lower level.

  1: Set RF receiving interrupt priority to higher level

6.3 Function

6.3.1 Interrupt Sources

If IE.EEW is set, write EEPROM finish can generates an interrupt, in the same time if EA is set the hardware will vector the process to the interrupt vector address of EEPROM write finish interrupt.

If EIE.ERFR is set, receive a byte from RF can generates an interrupt, in the same time if EA is set the hardware will vector the process to the interrupt vector address of RF receiving interrupt.

Each of the individual interrupts can be enabled or disabled by setting or clearing a bit in the IE/EIE SFR. IE also has a global enable/disable bit EA, which can be cleared to disable all the interrupts.

6.3.2 Priority Level Structure

There are two priority levels for the interrupts: high and low. The interrupt sources can be individually set to either high or low levels. Naturally, a higher priority interrupt cannot be interrupted by a lower priority interrupt. However there exists a predefined hierarchy amongst the interrupts themselves. This hierarchy comes into play when the interrupt controller has to resolve simultaneous requests having the same priority level. This hierarchy is defined as shown below, the interrupts are numbered starting from the highest priority to the lowest.

Table 6-5 Priority structure of interrupts

Interrupt request	Priority		Vector
EEW(high IP.2=1)	1	High level	13H
ERFR (high EIP.0 =1)	2		43H
EEW(low IP.2=0)	3	Low level	13H

Interrupt request	Priority		Vector
ERFR (low EIP.0 =0)	4		43H

The interrupt flags are sampled in every machine cycle. In the same machine cycle, the sampled interrupts are polled and their priority is resolved. If an interrupt is detected, and the certain conditions are met, the hardware will execute an internally generated LCALL instruction, which will vector the process to the appropriate interrupt vector address. The conditions for generating the LCALL are:

- An interrupt of equal or higher priority is not currently being serviced.
- The current polling cycle is the last machine cycle of the instruction being executed currently.
- The current instruction does not involve a write to IP, IE, EIP or EIE registers and is not a RETI.

The polling cycle is repeated every machine cycle, with the interrupts sampled in the same machine cycle. The processor responds to a valid interrupts by executing an LCALL instruction to the appropriate service routine. This may or may not clear the flag, which caused the interrupt.

In case of EEPROM write finish interrupt, the interrupt flag will be cleared by hardware automatically whenever the processor vectors to the EEPROM write finish interrupt service routine.

The RF receiving interrupt flag, BUF_CON.RDATA is only controlled by RF block itself. When a complete frame is received, this bit is set by hardware. It should be cleared by software to enable next receiving.

The hardware LCALL behaves exactly like the software LCALL instruction. This instruction saves the Program Counter contents onto the Stack, but does not save the Program Status Word PSW. The PC is reloaded with the vector address of that interrupt which caused the LCALL.

The vector table is not evenly spacing, this is to accommodate future expansions to the device family. Execution continues from the vectored address till an RETI instruction is executed. On execution of the RETI instruction the processor pops the Stack and loads the PC with the contents from the top of the stack. The user must take care that the status of the stack is restored to what it was after the hardware LCALL, if the execution is to return to the interrupted program. The processor does not notice anything if the stack contents are modified and will proceed with execution from the address put back into PC. Note that a RET instruction would do exactly the same process as a RETI instruction, but it would not inform the Interrupt Controller, that the interrupt service routine is completed, and would leave the controller still thinking that the service routine is under progress.

Note:

When enable the interrupt, 2 NOP instructions should be added following the interrupt enabling instruction.

7 Memory

7.1 EEPROM

The size of EEPROM is 8K bytes, address range: E000~FFFFH. The first 1K bytes (E000~E3FFH) of EEPROM is for Mifare1 data memory (E000~E00F is read only). The rest part of EEPROM is general data memory and can be used as code memory. MOVC from this block is inhabited. It can’t access the COS area (24K ROM/EEPROM area used as code memory) by MOVC.

The CPU can access all the 8K bytes of EEPROM.

7.1.1 Mifare1 EEPROM

The 1K bytes EEPROM is configured for Mifare1 mode. The CPU should be responsible for the data format in this area and the hardware will not check the data format in CPU mode.

7.1.2 Registers

7.1.2.1 EE_PGM_CON (97H)

Table 7-1 EEPROM Control Register (EE_PGM_CON)

EE_PGM_CON	7	6	5	4	3	2	1	0
BIT	LOCK	RSV	RSV	RSV	RSV	RSV	RSV	PGM
RST	0	0	0	0	0	0	0	0
W/R	W/R	R	R	R	R	R	R	W/R

- LOCK

0: No effect

1: EEPROM program is inhibited.

- PGM:

0: Program EEPROM is complete, this bit can not cleared by software and is cleared by hardware automatically

1: Start program EEPROM, the data that has been written into buffer is programmed into the EEPROM array.

Note:

Cross page writing is not supported.

When RF in transmission mode, EEPROM program should start after the transmission is done.

7.2 IRAM

The internal SRAM size is 256 bytes (00~FFH).

7.3 XRAM

The external SRAM size is 1K bytes (0000~02FFH, 3000~30FFH). The higher 256 bytes is RF buffer.

7.4 ROM

The User ROM/EEPROM is 24K bytes (0000~5FFFH).

8 CIU

8.1 Overview

CIU is a basic Mifare1 controller which is fully compatible with Mifare1 S50.

8.2 Mifare1 Memory Organization

1K bytes EEPROM are organized in 16 sectors with 4 blocks and each block contains 16 bytes. The first 3 blocks are data blocks. The last block is "sector trailer", which contains 2 keys and access condition for each sector.
The block 0 of sector 0 is vendor data and is read only, they are defined as follow:

Table 8-1 contents of Sector 0 Block 0Value Block

B0	B1	B2	B3	B4	B5	B6	B7
UID0	UID1	UID2	UID3	Check Byte	SAK	ATQA	
B8	B9	B10	B11	B12	B13	B14	B15
MID	Reserved	Reserved	Reserved	MAC0	MAC1	MAC2	MAC3

The definition of other blocks is same as Mifare1 S50.

8.3 High Baudrate Support

In CPU mode, RF interface could receive/send 106k/212k bit rate data from/to PCD. The bit rate is controlled by SFR bit: rcvbaud[1:0] and sndbaud[1:0].

When rcvbaud is set to 01, RF interface should receive 212k bit rate data, otherwise, it should receive 106k bit rate data.

When sndbaud is set to 01, RF interface should send 212k bit rate data, otherwise, it should send 106k bit rate data.

Note: CPU mode support both 106k and 212k bit rate, and CIU mode support 106k bit rate only.

Table 8-2 RF interface Control (BAUDSEL)

BAUDSEL	7	6	5	4	3	2	1	0
BIT	RCVBAUD1	RCVBAUD0	SNDBAUD1	SNDBAUD0	SWITCH	RSV	RSV	RSV
RST	0	0	0	0	0	0	0	0
W/R	W/R	W/R	W/R	W/R	W/R	W/R	W/R	W/R

- RCVBAUD1, RCVBAUD0:

  Receive baudrate select.

  00: 106k bit rate

  01: 212k bit rate

  10: reserved.

  11: reserved.

- SNDBAUD1,SNDBAUD0:

00:	106k bit rate
01:	212k bit rate
10:	reserved.
11:	reserved.

- SWITCH:

  Baudrate Switch Point:

  0: RF switch the baudrate after sending the response data at previous baudrate

  1: RF switch the baudrate immediately

9 CIU Bridge with CPU (CBC)

9.1 Overview

CBC bridges the data and control between the CIU and CPU. A 256 Bytes XRAM (3000H~30FFH) is used as receiving or transmission buffer. A frame exceed 256 bytes will cause overflow. CPU can access this area when neither in receiving nor in transmission.

9.2 Registers

9.2.1 BUF_CON (ABH)

Table 9-1 Buffer Control (BUF_CON)

BUF_CON	7	6	5	4	3	2	1	0
BIT	DESEL_N	CPUOFF	HCRC_CEN	HCRC_GEN	RCV_PRT	RDATA	TBUF	RSV
RST	1	0	1	0	0	0	0	0
W/R	W/R	W/R	W/R	W/R	W/R	W/R	W/R	R

- DESEL_N:

  0: turn CPU mode off, Software should finish all jobs before clear this bit. Then the chip will enter HALT state and can be wakeup only by WUPA.

  1: turn CPU mode on, this bit can only be set by hardware after received RATS command.

- CPUOFF:

  0: turn CPU mode on, this bit can only be cleared by hardware after received RATS command.

  1: turn CPU mode off, Software should finish all jobs before set this bit. If the chip state is IDLE before entered CPU mode it will enter IDLE state, otherwise it will enter HALT state.

- HCRC_CEN:

  0: disable hardware CRC check

  1: enable hardware CRC check

- HCRC_GEN:

  0: disable hardware CRC generation

  1: enable hardware CRC generation

- RCV_PRT: Buffer protection when chip is in receiving mode.

  1: enable RF buffer protection, to protect the RF buffer, BUF_SIZE and BIT_CNT from being changed by RF interface when BUF_CON.RDATA is set.

  0: disable RF buffer protection

- RDATA:

  1: this bit is automatically set by hardware once RF received a frame data. If EIE.ERFR is set a RF receiving interrupt will be activated.

0: software should clear this bit to enable next receiving.

- TBUF:

  1: set by software to start RF transmission

  0: this bit is cleared by hardware after transmission

Note:

Clear the DESEL_N or set the CPUOFF can close the CPU within 1 ETU.

9.2.2 BUF_STATUS (ACH)

Table 9-2 Buffer Status (BUF_STATUS)

BUF_STATUS	7	6	5	4	3	2	1	0
BIT	NACK	RSV	RSV	RSV	RSV	CRCERR	PERR	OVL
RST	0	0	0	0	0	0	0	0
W/R	W/R	R	R	R	R	W/R	W/R	W/R

- NACK:

  Set to switch IS23SC4456 to receive mode, set by software when commands from PCD needs no response, this bit is automatically cleared by hardware

  NOTE: If BAUDSEL[7:4] equals to "0000" this bit should NOT be set!!

- OVL: Overflow flag.

  1: overflow, set by hardware automatically when frame exceeds the RF buffer size

  0: no overflow

- PERR: Parity error flag.

  1: parity error, set by hardware automatically

  0: no parity error

- CRCERR: CRC error flag, only valid when BUF_CON.HCRC_CEN is set.

  1: CRC error, set by hardware automatically

  0: no CRC error

9.2.3 BUF_SIZE (ADH)

Table 9-3 Buffer Size (BUF_SIZE)

BUF_SIZE	7	6	5	4	3	2	1	0
BIT	D7	D6	D5	D4	D3	D2	D1	D0
RST	0	0	0	0	0	0	0	0
W/R	W/R	W/R	W/R	W/R	W/R	W/R	W/R	W/R

In receiving mode, the buffer can receive maximum 256bytes data. A frame input exceeding 256bytes will cause overflow. If D7~D0 is 0 and RDATA is set, then there is 256bytes received in buffer.

The starting address of buffer address is 3000H. And the D7~D0 indicate the size of current receiving/transmission frame.

9.2.4 BIT_CNT (AEH)

Table 9-4 Bit Counter (BIT_CNT)

BIT_CNT	7	6	5	4	3	2	1	0
BIT	RSV	RSV	RSV	RSV	RSV	D2	D1	D0
RST	0	0	0	0	0	0	0	0
W/R	R	R	R	R	R	W/R	W/R	W/R

- D2:D0:

  The bit count received/sending. 0~7bit is support.

  If D2:D0 is 00, no extra bit is received/sending. This register is only valid for the last receiving or transmitting byte.

9.3 Function

9.3.1 Data Receiving

In Mifare1 mode, the RF buffer is turned off. The data is read into CIU directly. After the CPU mode is activated by hardware, the CPU communicates with the RF via RF buffer. And the data path to CIU is disabled.

Once a data frame is fully received, BUF_CON.RDATA is set. Hardware will check the parity error automatically, and set the BUF_STATUS.PERR accordingly.

If BUF_CON.RCV_PRT is set, software should clear BUF_CON.RDATA to enable next receiving, otherwise no data will be received into the RF buffer. If the BUF_CON.RCV_PRT is set, the BUF_CON.RDATA can be cleared by setting BUF_CON.TBUF.

9.3.2 Data transmission

Once BUF_CON.TBUF is set, hardware starts to transmit the data in the RF buffer. The data in buffer must be prepared first before transmission and CPU has no right to access to the XRAM if BUF_CON.TBUF is set. A frame contains data in partial byte is supported to transmit by configuring BIT_CNT.

During transmission, the RF buffer, BUF_SIZE and BIT_CNT are not accessible.

Hardware will automatically append the parity bit for transmission.

9.3.3 Hardware CRC

IS23SC4456 supports the hardware CRC check/generation during receiving and transmission if BUF_CON.HCRC_CEN/BUF_CON.HCRC_GEN is set.

9.3.4 CPU Mode

9.3.4.1 Activation of CPU Mode

CPU will be waken up once receiving RATS.

9.3.4.2 Exit CPU Mode

Software can either clear BUF_CON.DESEL_N or set BUF_CON.CPUOFF to exit CPU mode.

10 True Random Number Generator

10.1 Overview

There is a true random number generator in IS23SC4456.

10.2 Register

10.2.1 RNGD（C6H）

Table 10-1: Random Number Generator Data register (RNGD)

RNGD	7	6	5	4	3	2	1	0
BIT	RNGD.7	RNGD.6	RNGD.5	RNGD.4	RNGD.3	RNGD.2	RNGD.1	RNGD.0
RST	0/1	0/1	0/1	0/1	0/1	0/1	0/1	0/1
W/R	R	R	R	R	R	R	R	R

Software can get random number through reading RNGD.

The RNG needs 8 CPU clock cycles to generate an 8-bit random number. After reading 1 byte from RNGD, the software must wait enough time to have the internal registers refreshed. If the CPU is run in 1/2 external clock frequency, it is better to read every one byte data from RNGD every 16 CPU clock cycles.

11 CRC Coprocessor

11.1 Overview

IS23SC4456 has 16 bits hardware CRC coprocessor, which implements the CRC generation polynomial (x^16+x^12+x^5+1) recommended by CCITT. It can be accessed by the CPU through SFRs to generate a CRC checksum value.

11.2 Register

11.2.1 CRC_CON (CEH)

Table 11-1 CRC Control Registers (CRC_CON)

CRC_CON	7	6	5	4	3	2	1	0
BIT	RSV	RSV	RSV	RSV	RSV	RECORDCLR	CRCREV	CRCLD
RST	0	0	0	0	0	0	0	0
W/R	R	R	R	R	R	W/R	W/R	W/R

- RECORDCLR

  0: No effect

  1: reset CRC block

  RECORDCLR is cleared by hardware automatically.

- CRCREV

  0: The content of internal CRC register is read by CPU directly

  1: The content of internal CRC register is reversed, when read by CPU

  The value of CRCREV does not affect the result of writing CRC_DAT.

- CRCLD

  0: The data written into CRC_DAT is used to calculate the CRC checksum

  1: The data written into CRC_DAT is loaded into internal CRC register directly

11.2.2 CRC_DAT (CFH)

Table 11-2 CRC Data Registers (CRC_DAT)

CRC_DAT	7	6	5	4	3	2	1	0
BIT	D7	D6	D5	D4	D3	D2	D1	D0
RST	0	0	0	0	0	0	0	0
W/R	W/R	W/R	W/R	W/R	W/R	W/R	W/R	W/R

If CRCCON.CRCLD is set, the data written into CRCDAT will be loaded into the internal 16bit CRC registers directly. If the CRCCON.CRCLD is clear, the data written will be used to calculate the CRC checksum.

If the CRCCON.REVERSE is set, reading the CRCDAT will get the reverse of the content in the 16-bit CRC registers. The first reading CRCDAT gets the b0~b7 of the internal CRC register, the second reading gets the b8~b15 of the internal CRC register. If the CRCCON.REVERSE is clear, reading the CRC_DAT will get the content of the CRC register directly. The first reading CRC_DAT gets the b7~b0 of the internal CRC register, the second reading gets the b15~b8 of the internal CRC register.

11.3 Function

11.3.1 CRC initialization

Write CRC_CON with 05H to set CRC load data mode, and clear access record. (The CRC_CON.RECORDCLR is cleared automatically)

Write data0 to CRC_DAT, the data in b7~b0 is shifted into b15~b8, the data0 written into CRC_DAT will be loaded into b7~b0. Write data1 to CRC_DAT again, the data0 in b7~b0 is shifted into b15~b8, the data1 written into CRC_DAT will be loaded into b7~b0. As a result, the data0 is loaded into b15~b8 of internal CRC register, the data1 is loaded into b7~b0 of the internal CRC register.

11.3.2 CRC calculation

Clear CRC_CON, then write data to CRC_DAT, the data written into CRC will be used to calculate the checksum.

11.3.3 Read CRC checksum

Set CRC_CON.RECORDCLR to clear access record, then read CRC_DAT to get the lower 8bits of internal register, then read CRC_DAT to get the higher 8bits of internal register. If CRC_CON.CRCREV is cleared, the 1st reading of CRC_DAT gets the lower 8 bits value of the internal register, the 2nd reading of CRC_DAT gets the higher 8 bits value of the internal register. If CRC_CON.CRCREV is set, the 1st reading of CRC_DAT gets the reverse of the lower 8 bits value, the 2nd gets the reverse of the higher 8 bits value.

12 DES Coprocessor

12.1 Overview

The DES coprocessor performs either encryption calculations according to the Data Encryption Standard (DES) or calculation in the well-known Triple-DES mode (TDES).

The DES hardware block performs a single DES en- or decryption in 32 clock cycles, a Triple-DES operation in 96 clock cycles. Additional CPU cycles are needed for software loading KEY and DATA into the DES block in advance.

Software can set the key and data (plain/cipher). The coprocessor will encrypt/decrypt the data according to the setting of register DES_CON, the following function can be done by the DES coprocessor

- Single DES Encrypt
- Single DES Decrypt
- Triple DES Encrypt with the sequence: encrypt with KEYA, decrypt with KEYB, encrypt with KEYA
- Triple DES decrypt with the sequence: decrypt with KEYA, encrypt with KEYB, decrypt with KEYA

12.2 Register

12.2.1 DES_CON (DDH)

Table 12-1 DES Control Registers (DES_CON)

DES_CON	7	6	5	4	3	2	1	0
BIT	GO	CNTRST	RSV	RSV	RSV	KBSEL	TDES	DESDEC
RST	0	0	0	0	0	0	0	0
W/R	W/R	W/R	R	R	R	W/R	W/R	W/R

- GO:

  0: No effect

  1: Start DES operation

  Start run DES operation, this bit shall be set by software, after DES operation, DES_CON.GO will be cleared by hardware automatically.

  Once the DES_CON.GO is set and it has not been cleared by hardware, the value of DES_CON, DES_KEY, and DES_DAT can not be changed by software. Writing these registers has no effect.

- CNTRST

  0: No effect.

  1: Reset internal counter

An internal counter records the number of DES_KEY and DES_DAT. Setting DES_CON.CNTRST can clear the counter. This bit is cleared by hardware automatically.

- KBSEL

  0: The data written into DESKEY register is KEYA (internal key-registers), The single DES operates with KEYA.

  1: The data written into DESKEY register is KEYB (internal key-registers), The single DES operates with KEYB.

- TDES:

  0: Single DES

  1: Triple DES

  Single DES needs 32 system clocks, TDES needs 96 system clocks.

- DESDEC

  0: Encrypt operation

  1: Decrypt operation, Single DES run the decrypt

  If this bit is cleared, then DES coprocessor work on encrypt mode, DES coprocessor will use keyA-keyB-keyA as operation key for triple DES. If this bit is set, then DES coprocessor works on decrypt mode, DES coprocessor will use keyA-keyB-keyA as operation key for triple DES

12.2.2 DES_KEY (DEH)

Table 12-2 DES Key Registers (DES_KEY)

DES_KEY	7	6	5	4	3	2	1	0
BIT	D7	D6	D5	D4	D3	D2	D1	D0
RST	0	0	0	0	0	0	0	0
W/R	W	W	W	W	W	W	W	W

The coprocessor contains 16 bytes internal key-registers to hold the key. The 16 bytes internal key-registers are divided into two parts. One part holds the KEYA, the other holds the KEYB.

12.2.3 DES_DAT (DFH)

Table 12-3 DES Data Register (DES_DAT)

DES_DAT	7	6	5	4	3	2	1	0
BIT	D7	D6	D5	D4	D3	D2	D1	D0
RST	0	0	0	0	0	0	0	0
W/R	W/R	W/R	W/R	W/R	W/R	W/R	W/R	W/R

12.3 Function

12.3.1 Load Key

DES block contains two 64-bit internal key-registers as KEYA and KEYB .To load keys to the internal key-registers, 8 consecutive bytes must be written into DES_KEY, the hardware load the data into KEYA or KEYB according to the DESCON.KBSEL.

12.3.2 Write/Read Data

DES block contains a 64-bit internal data-register. Before set the DES_CON.GO, the software must write plain data or cipher data to the DES_DAT. The data written to DES_DAT is loaded into the internal 64-bit data register, 8 consecutive bytes must be written into DES_DAT to fill with the internal 64-bit data register. The data read from DES_DAT is from the internal 64-bit data register, 8 consecutive bytes must be read out.

13 POWER Management

13.1 Overview

Low power design is very important for contact-less chip. There are efficient features to do power management.

13.2 Registers

13.2.1 SYS_CLK_CON (A6H)

Table 13-1 Clock Control (SYS_CLK_CON)

SYS_CLK_CON	7	6	5	4	3	2	1	0
BIT	RSV	RSV	RSV	RSV	D3	D2	D1	D0
RST	0	0	0	0	0	0	1	0
W/R	R	R	R	R	W/R	W/R	W/R	W/R

- D3:D0:

  Adjust the system clock for MCU.

  0001: 1/2 of External frequency: 6.78 MHz

  0010: 1/4 of External frequency: 3.39 MHz

  Others: reserved

If write the SFR with other value, then the return value will be 02H, and chip use the 1/4 of external frequency as the system clock.

13.2.2 PCON (87H)

Table 13-2 Power Control register (PCON)

PCON	7	6	5	4	3	2	1	0
BIT	RSV	RSV	RSV	RSV	GF1	GF0	PD	IDL
RST	0	0	1	1	0	0	0	0
W/R	W/R	W/R	R	R	W/R	W/R	W/R	W/R

- GF1-0

  These two bits are general-purpose user flags.

- PD

  0: No effect

  1: Set CPU to SLEEP mode, all the clocks are stopped and program execution is frozen. The SLEEP mode can be terminated by EIE.FEW (EEPROM write finish) or BUF_CON.RDATA (RF receiving)

Note:

2 NOP instructions must be added after the instruction of setting the PD bit

- IDL

  0: No effect

  1: Set CPU to IDLE mode, clock to CPU is stopped and program execution is frozen. But clock to RF block, interrupt and other peripheral blocks is still active. The IDL mode can exit by interrupt.

13.2.3 EPCON (8FH)

Table 13-3 Extended Power Control Register (EPCON)

EPCON	7	6	5	4	3	2	1	0
BIT	RSV	RSV	RSV	RSV	CRC	RNG	DES	RSV
RST	0	0	0	0	0	0	0	0
W/R	W/R	W/R	R	R	W/R	W/R	W/R	W/R

- CRC

  0: Switch off CRC

  1: Switch on CRC

- RNG

  0: Switch off random number generator

  1: Switch on random number generator

- DES

  0: Switch off DES

  1: Switch on DES

14 Security Feature

14.1 Overview

There are four kinds of security logic provided:

- Voltage sensor
- Frequency sensor
- Memory encryption
- MOVC block

14.2 Register

14.2.1 SECCON (9EH)

Table 14-1 Security Control Register (SECCON)

SECCON	7	6	5	4	3	2	1	0
BIT	RSV	RSV	RSV	RSV	PORF	RSV	RSV	VSENSOR
RST	0	0	0	1	1	0	1	1
W/R	R	R	W/R	R	W/R	R	R	W/R

- PORF, Power-on reset flag:

  Hardware will set this flag on a power up condition. This flag can be read or written by software. Writing 0 to this bit is the only way to clear this bit once it is set. This bit can only be written 0; writing 1 to this bit has no effect.

- VSENOR

  0: Switch off voltage sensor

  1: Switch on voltage sensor

14.3 Sensor

14.3.1 High Frequency Filter

Internal system clock frequency is consistently monitored.

14.3.2 Low Voltage Sensor

Internal VDC output voltage is consistently monitored. If it is too low, a low voltage reset is asserted. Software can switch on/off voltage sensor.

14.4 Memory Encryption

Memory encryption is a feature that provides a very effective additional security layer.

- Data stored in ROM, EEPROM is encrypted
- Data transfer between memories is encrypted

- Encryption uses an unpublished proprietary symmetric block cipher
- Plain-text modifications cause unpredictable cipher text changes. This means that a single-bit cipher text change cannot be predicted from a specific plain text bit modification.

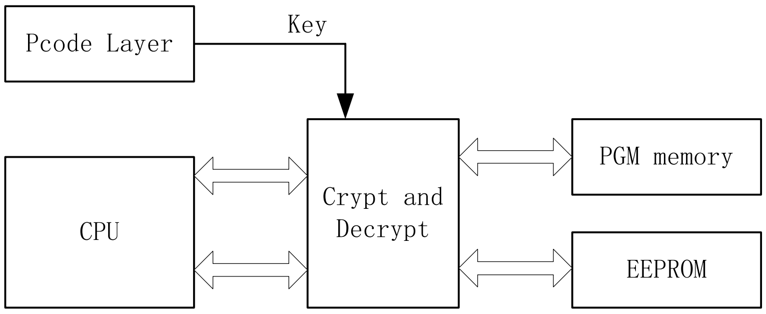

Figure 14-1 Memory Encryption

14.5 MOVC block

MOVC block is that code executed in 8K EEPROM does not have read access to 24K ROM/EEPROM. Once the MOVC is blocked, the data fetched by MOVC is 00H.

15 Characteristics

Table 15-1 Characteristics

PARAMETER	CONDITIONS	MIN.	TYP.	MAX.
Operating frequency		12.56MHz	13.56Mhz	14.56MHz
Input capacitance	22°C, 13.56MHz, 2V	14.40pf	15.90pf	17.4pf
ESD	HBM	6 kV		
EEPROM write time			3.0ms	5.0ms
EEPROM data retention		10 years		
EEPROM write endurance		100,000 cycles		
Working distance				10.0cm
Resonance frequency			16.0MHz	

Appendix A-------Instruction Set

The DEVICE executes all the instructions of the standard 8051 family. The operation of these instructions, their effect on the flag bits the status bits is exactly the same. However, timing of these instructions is different.

1. Flag setting

Table: Instructions that affect Flag settings

Instruction	Carry	Overflow	Auxiliary Carry	Instruction	Carry	Overflow	Auxiliary Carry
ADD	X	X	X	SETB C	1		
ADDC	X	X	X	CLR C	0		
SUBB	X	X	X	CPL C	X		
MUL	0	X		ANL C, bit	X		
DIV	0	X		ORL C, bit	X		
DA A	X			MOV C, bit	X		
RRC A	X			CJNE	X		
RLC A	X						

X indicates that the modification is as per the result of the instruction.

2. Instruction set

Table: Instruction Set for DEVICE

Op-code	HEX Code	Bytes	Machine Cycles	Clock cycles
NOP	00	1	1	4
ADD A, R0	28	1	1	4
ADD A, R1	29	1	1	4
ADD A, R2	2A	1	1	4
ADD A, R3	2B	1	1	4
ADD A, R4	2C	1	1	4
ADD A, R5	2D	1	1	4
ADD A, R6	2E	1	1	4
ADD A, R7	2F	1	1	4
ADD A, @R0	26	1	1	4
ADD A, @R1	27	1	1	4
ADD A, direct	25	2	2	8
ADD A, #data	24	2	2	8
ADDC A, R0	38	1	1	4
ADDC A, R1	39	1	1	4
ADDC A, R2	3A	1	1	4
ADDC A, R3	3B	1	1	4
ADDC A, R4	3C	1	1	4
ADDC A, R5	3D	1	1	4
ADDC A, R6	3E	1	1	4
ADDC A, R7	3F	1	1	4
ADDC A, @R0	36	1	1	4
ADDC A, @R1	37	1	1	4
ADDC A, direct	35	2	2	8
ADDC A, #data	34	2	2	8
SUBB A, R0	98	1	1	4

Op-code	HEX Code	Bytes	Machine Cycles	Clock cycles
SUBB A, R1	99	1	1	4
SUBB A, R2	9A	1	1	4
SUBB A, R3	9B	1	1	4
SUBB A, R4	9C	1	1	4
SUBB A, R5	9D	1	1	4
SUBB A, R6	9E	1	1	4
SUBB A, R7	9F	1	1	4
SUBB A, @R0	96	1	1	4
SUBB A, @R1	97	1	1	4
SUBB A, direct	95	2	2	8
SUBB A, #data	94	2	2	8
INC A	04	1	1	4
INC R0	08	1	1	4
INC R1	09	1	1	4
INC R2	0A	1	1	4
INC R3	0B	1	1	4
INC R4	0C	1	1	4
INC R5	0D	1	1	4
INC R6	0E	1	1	4
INC R7	0F	1	1	4
INC @R0	06	1	1	4
INC @R1	07	1	1	4
INC direct	05	2	2	8
INC DPTR	A3	1	3	12
DEC A	14	1	1	4
DEC R0	18	1	1	4
DEC R1	19	1	1	4
DEC R2	1A	1	1	4
DEC R3	1B	1	1	4
DEC R4	1C	1	1	4
DEC R5	1D	1	1	4
DEC R6	1E	1	1	4
DEC R7	1F	1	1	4
DEC @R0	16	1	1	4
DEC @R1	17	1	1	4
DEC direct	15	2	2	8
DEC DPTR	A5	1	3	12
MUL AB	A4	1	5	20
DIV AB	84	1	5	20
DA A	D4	1	1	4
ANL A, R0	58	1	1	4
ANL A, R1	59	1	1	4
ANL A, R2	5A	1	1	4
ANL A, R3	5B	1	1	4
ANL A, R4	5C	1	1	4
ANL A, R5	5D	1	1	4
ANL A, R6	5E	1	1	4
ANL A, R7	5F	1	1	4
ANL A, @R0	56	1	1	4
ANL A, @R1	57	1	1	4
ANL A, direct	55	2	2	8

Op-code	HEX Code	Bytes	Machine Cycles	Clock cycles
ANL A, #data	54	2	2	8
ANL direct, A	52	2	2	8
ANL direct, #data	53	3	3	12
ORL A, R0	48	1	1	4
ORL A, R1	49	1	1	4
ORL A, R2	4A	1	1	4
ORL A, R3	4B	1	1	4
ORL A, R4	4C	1	1	4
ORL A, R5	4D	1	1	4
ORL A, R6	4E	1	1	4
ORL A, R7	4F	1	1	4
ORL A, @R0	46	1	1	4
ORL A, @R1	47	1	1	4
ORL A, direct	45	2	2	8
ORL A, #data	44	2	2	8
ORL direct, A	42	2	2	8
ORL direct, #data	43	3	3	12
XRL A, R0	68	1	1	4
XRL A, R1	69	1	1	4
XRL A, R2	6A	1	1	4
XRL A, R3	6B	1	1	4
XRL A, R4	6C	1	1	4
XRL A, R5	6D	1	1	4
XRL A, R6	6E	1	1	4
XRL A, R7	6F	1	1	4
XRL A, @R0	66	1	1	4
XRL A, @R1	67	1	1	4
XRL A, direct	65	2	2	8
XRL A, #data	64	2	2	8
XRL direct, A	62	2	2	8
XRL direct, #data	63	3	3	12
CLR A	E4	1	1	4
CPL A	F4	1	1	4
RL A	23	1	1	4
RLC A	33	1	1	4
RR A	03	1	1	4
RRC A	13	1	1	4
SWAP A	C4	1	1	4
MOV A, R0	E8	1	1	4
MOV A, R1	E9	1	1	4
MOV A, R2	EA	1	1	4
MOV A, R3	EB	1	1	4
MOV A, R4	EC	1	1	4
MOV A, R5	ED	1	1	4
MOV A, R6	EE	1	1	4
MOV A, R7	EF	1	1	4
MOV A, @R0	E6	1	1	4
MOV A, @R1	E7	1	1	4
MOV A, direct	E5	2	2	8
MOV A, #data	74	2	2	8
MOV R0, A	F8	1	1	4

Op-code	HEX Code	Bytes	Machine Cycles	Clock cycles
MOV R1, A	F9	1	1	4
MOV R2, A	FA	1	1	4
MOV R3, A	FB	1	1	4
MOV R4, A	FC	1	1	4
MOV R5, A	FD	1	1	4
MOV R6, A	FE	1	1	4
MOV R7, A	FF	1	1	4
MOV R0, direct	A8	2	2	8
MOV R1, direct	A9	2	2	8
MOV R2, direct	AA	2	2	8
MOV R3, direct	AB	2	2	8
MOV R4, direct	AC	2	2	8
MOV R5, direct	AD	2	2	8
MOV R6, direct	AE	2	2	8
MOV R7, direct	AF	2	2	8
MOV R0, #data	78	2	2	8
MOV R1, #data	79	2	2	8
MOV R2, #data	7A	2	2	8
MOV R3, #data	7B	2	2	8
MOV R4, #data	7C	2	2	8
MOV R5, #data	7D	2	2	8
MOV R6, #data	7E	2	2	8
MOV R7, #data	7F	2	2	8
MOV @R0, A	F6	1	1	4
MOV @R1, A	F7	1	1	4
MOV @R0, direct	A6	2	2	8
MOV @R1, direct	A7	2	2	8
MOV @R0, #data	76	2	2	8
MOV @R1, #data	77	2	2	8
MOV direct, A	F5	2	2	8
MOV direct, R0	88	2	2	8
MOV direct, R1	89	2	2	8
MOV direct, R2	8A	2	2	8
MOV direct, R3	8B	2	2	8
MOV direct, R4	8C	2	2	8
MOV direct, R5	8D	2	2	8
MOV direct, R6	8E	2	2	8
MOV direct, R7	8F	2	2	8
MOV direct, @R0	86	2	2	8
MOV direct, @R1	87	2	2	8
MOV direct, direct	85	3	3	12
MOV direct, #data	75	3	3	12
MOV DPTR, #data 24	90	4	4	16
MOVC A, @A+DPTR	93	1	3	12
MOVC A, @A+PC	83	1	3	12
MOVX A, @R0	E2	1	2-9	8-36
MOVX A, @R1	E3	1	2-9	8-36
MOVX A, @DPTR	E0	1	2-9	8-36
MOVX @R0, A	F2	1	2-9	8-36
MOVX @R1, A	F3	1	2-9	8-36
MOVX @DPTR, A	F0	1	2-9	8-36

Op-code	HEX Code	Bytes	Machine Cycles	Clock cycles
PUSH direct	C0	2	2	8
POP direct	D0	2	2	8
XCH A, R0	C8	1	1	4
XCH A, R1	C9	1	1	4
XCH A, R2	CA	1	1	4
XCH A, R3	CB	1	1	4
XCH A, R4	CC	1	1	4
XCH A, R5	CD	1	1	4
XCH A, R6	CE	1	1	4
XCH A, R7	CF	1	1	4
XCH A, @R0	C6	1	1	4
XCH A, @R1	C7	1	1	4
XCHD A, @R0	D6	1	1	4
XCHD A, @R1	D7	1	1	4
XCH A, direct	C5	2	2	8
CLR C	C3	1	1	4
CLR bit	C2	2	2	8
SETB C	D3	1	1	4
SETB bit	D2	2	2	8
CPL C	B3	1	1	4
CPL bit	B2	2	2	8
ANL C, bit	82	2	2	8
ANL C, /bit	B0	2	2	8
ORL C, bit	72	2	2	8
ORL C, /bit	A0	2	2	8
MOV C, bit	A2	2	2	8
MOV bit, C	92	2	2	8
ACALL addr19	11,31,51, 71, 91,B1, D1, F1	3	4	16
LCALL addr24	12	4	5	20
RET	22	1	3	12
RETI	32	1	3	12
AJMP ADDR19	01,21,41, 61,81,A1, C1, E1	3	4	16
LJMP addr24	02	4	5	20
JMP @A+DPTR	73	1	3	12
SJMP rel	80	2	3	12
JZ rel	60	2	3	12
JNZ rel	70	2	3	12
JC rel	40	2	3	12
JNC rel	50	2	3	12
JB bit, rel	20	3	4	16
JNB bit, rel	30	3	4	16
JBC bit, rel	10	3	4	16
CJNE A, direct, rel	B5	3	4	16
CJNE A, #data, rel	B4	3	4	16
CJNE @R0, #data, rel	B6	3	4	16
CJNE @R1, #data, rel	B7	3	4	16
CJNE R0, #data, rel	B8	3	4	16
CJNE R1, #data, rel	B9	3	4	16
CJNE R2, #data, rel	BA	3	4	16
CJNE R3, #data, rel	BB	3	4	16
CJNE R4, #data, rel	BC	3	4	16

Op-code	HEX Code	Bytes	Machine Cycles	Clock cycles
CJNE R5, #data, rel	BD	3	4	16
CJNE R6, #data, rel	BE	3	4	16
CJNE R7, #data, rel	BF	3	4	16
DJNZ R0, rel	D8	2	3	12
DJNZ R1, rel	D9	2	3	12
DJNZ R2, rel	DA	2	3	12
DJNZ R3, rel	DB	2	3	12
DJNZ R4, rel	DC	2	3	12
DJNZ R5, rel	DD	2	3	12
DJNZ R6, rel	DE	2	3	12
DJNZ R7, rel	DF	2	3	12
DJNZ direct, rel	D5	3	4	16