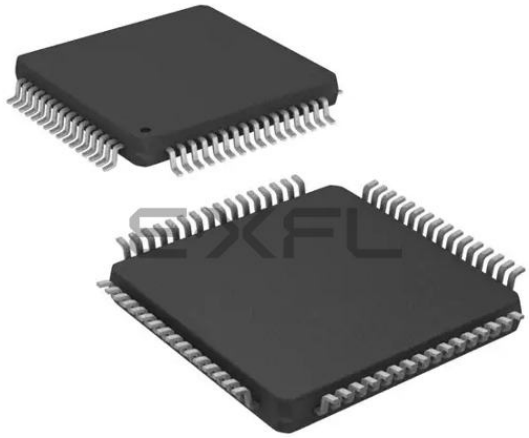


Welcome to E-XFL.COM

What is "Embedded - Microcontrollers"?

"Embedded - Microcontrollers" refer to small, integrated circuits designed to perform specific tasks within larger systems. These microcontrollers are essentially compact computers on a single chip, containing a processor core, memory, and programmable input/output peripherals. They are called "embedded" because they are embedded within electronic devices to control various functions, rather than serving as standalone computers. Microcontrollers are crucial in modern electronics, providing the intelligence and control needed for a wide range of applications.

Applications of "Embedded - Microcontrollers"

Details	
Product Status	Obsolete
Core Processor	ST7
Core Size	8-Bit
Speed	8MHz
Connectivity	SmartCard, UART/USART, USB
Peripherals	LED, LVD, POR, WDT
Number of I/O	35
Program Memory Size	16KB (16K x 8)
Program Memory Type	FLASH
EEPROM Size	-
RAM Size	768 x 8
Voltage - Supply (Vcc/Vdd)	4V ~ 5.5V
Data Converters	-
Oscillator Type	Internal
Operating Temperature	0°C ~ 70°C (TA)
Mounting Type	Surface Mount
Package / Case	64-LQFP
Supplier Device Package	-
Purchase URL	https://www.e-xfl.com/product-detail/stmicroelectronics/st7fscr1r4t1

4.8 Register description

FLASH control/status register (FCSR)

Read/Write

Reset Value: 0000 0000 (00h)

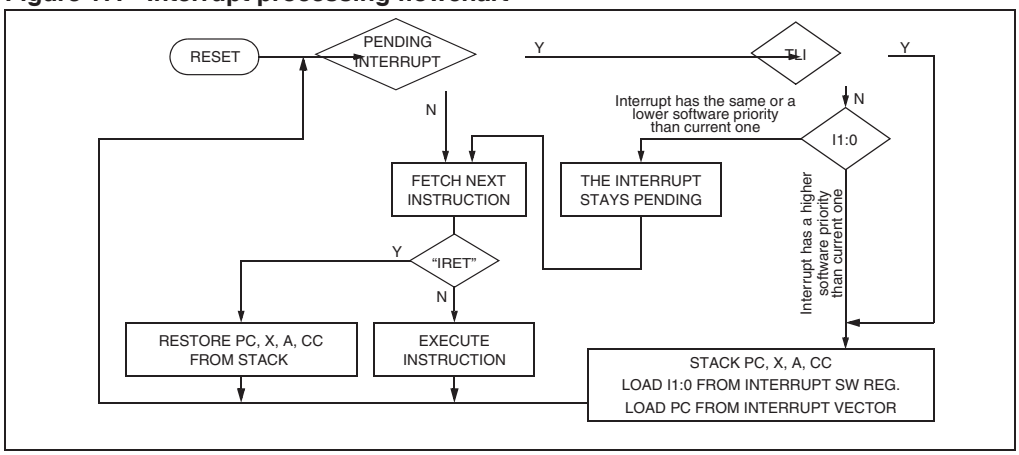
7							0
0	0	0	0	0	0	0	0

This register is reserved for use by Programming Tool software. It controls the FLASH programming and erasing operations. For details on customizing FLASH programming methods and In-Circuit Testing, refer to the ST7 FLASH Programming and ICC Reference Manual.

Table 7. Interrupt software priority levels

Interrupt software priority	Level	I1	I0
Level 0 (main)	Low ↓ High	1	0
Level 1		0	1
Level 2		0	0
Level 3 (= interrupt disable)		1	1

Figure 17. Interrupt processing flowchart



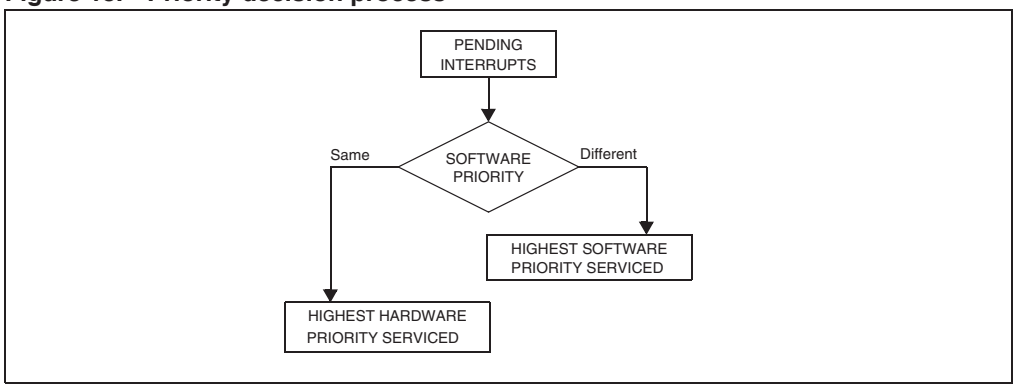
Servicing pending interrupts

As several interrupts can be pending at the same time, the interrupt to be taken into account is determined by the following two-step process:

- the highest software priority interrupt is serviced,
- if several interrupts have the same software priority then the interrupt with the highest hardware priority is serviced first.

Figure 18 describes this decision process.

Figure 18. Priority decision process



When an interrupt request is not serviced immediately, it is latched and then processed when its software priority combined with the hardware priority becomes the highest one.

Note: The hardware priority is exclusive while the software one is not. This allows the previous process to succeed with only one interrupt.

RESET, TRAP and TLI can be considered as having the highest software priority in the decision process.

Different interrupt vector sources

Two interrupt source types are managed by the CPU interrupt controller: the non-maskable type (RESET, TLI, TRAP) and the maskable type (external or from internal peripherals).

Non-maskable sources

These sources are processed regardless of the state of the I1 and I0 bits of the CC register (see *Figure 17*). After stacking the PC, X, A and CC registers (except for RESET), the corresponding vector is loaded in the PC register and the I1 and I0 bits of the CC are set to disable interrupts (level 3). These sources allow the processor to exit HALT mode.

- TLI (Top Level Hardware Interrupt)

This hardware interrupt occurs when a specific edge is detected on the dedicated TLI pin.

Caution: A TRAP instruction must not be used in a TLI service routine.

- TRAP (Non Maskable Software Interrupt)

This software interrupt is serviced when the TRAP instruction is executed. It will be serviced according to the flowchart in *Figure 17* as a TLI.

Caution: TRAP can be interrupted by a TLI.

- RESET

The RESET source has the highest priority in the CPU. This means that the first current routine has the highest software priority (level 3) and the highest hardware priority.
See the RESET chapter for more details.

Maskable sources

Maskable interrupt vector sources can be serviced if the corresponding interrupt is enabled and if its own interrupt software priority (in ISPRx registers) is higher than the one currently being serviced (I1 and I0 in CC register). If any of these two conditions is false, the interrupt is latched and thus remains pending.

- External Interrupts

External interrupts allow the processor to exit from HALT low power mode.
External interrupt sensitivity is software selectable through the register.
External interrupt triggered on edge will be latched and the interrupt request automatically cleared upon entering the interrupt service routine.
If several input pins of a group connected to the same interrupt line are selected simultaneously, these will be logically NANDed.

- Peripheral Interrupts

Usually the peripheral interrupts cause the Device to exit from HALT mode except those mentioned in the “Interrupt Mapping” table.
A peripheral interrupt occurs when a specific flag is set in the peripheral status registers and if the corresponding enable bit is set in the peripheral control register.
The general sequence for clearing an interrupt is based on an access to the status register followed by a read or write to an associated register.

Note: The clearing sequence resets the internal latch. A pending interrupt (i.e. waiting for being serviced) will therefore be lost if the clear sequence is executed.

7.3 Interrupts and low power modes

All interrupts allow the processor to exit the WAIT low power mode. On the contrary, only external and other specified interrupts allow the processor to exit from the HALT modes (see column “Exit from HALT” in “Interrupt Mapping” table). When several pending interrupts are present while exiting HALT mode, the first one serviced can only be an interrupt with exit from HALT mode capability and it is selected through the same decision process shown in *Figure 18*.

Note: If an interrupt, that is not able to Exit from HALT mode, is pending with the highest priority when exiting HALT mode, this interrupt is serviced after the first one serviced.

7.4 Concurrent and nested management

The following *Figure 19* and *Figure 20* show two different interrupt management modes. The first is called concurrent mode and does not allow an interrupt to be interrupted, unlike the nested mode in *Figure 20*. The interrupt hardware priority is given in this order from the lowest to the highest: MAIN, IT4, IT3, IT2, IT1, IT0, TLI. The software priority is given for each interrupt.

Warning: A stack overflow may occur without notifying the software of the failure.

Figure 19. Concurrent interrupt management

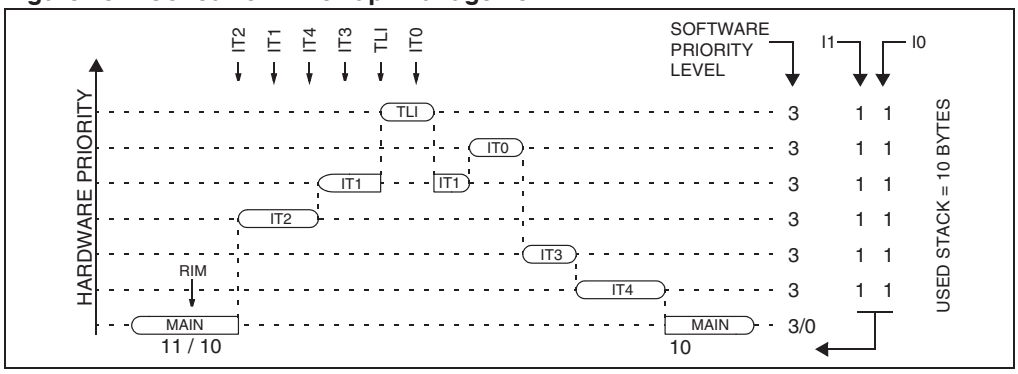
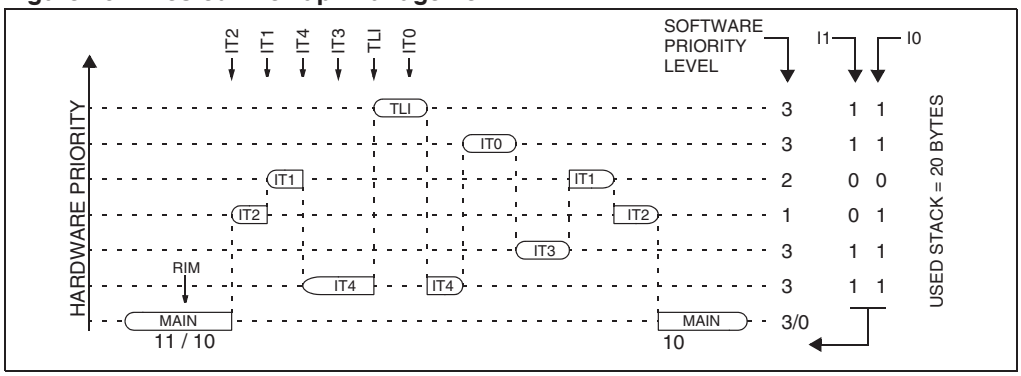


Figure 20. Nested interrupt management



7.5 Interrupt register description

CPU CC register interrupt bits

Read/Write

Reset Value: 111x 1010 (xAh)

7							0
1	1	I1	H	I0	N	Z	C

Bit 5, 3 = **I1, I0** *Software Interrupt Priority*

These two bits indicate the current interrupt software priority.

Table 8. Current interrupt software priority

Interrupt software priority	Level	I1	I0
Level 0 (main)	Low ↓ High	1	0
Level 1		0	1
Level 2		0	0
Level 3 (= interrupt disable*)		1	1

These two bits are set/cleared by hardware when entering in interrupt. The loaded value is given by the corresponding bits in the interrupt software priority registers (ISPRx).

They can be also set/cleared by software with the RIM, SIM, HALT, WFI, IRET and PUSH/POP instructions (see “Interrupt Dedicated Instruction Set” table).

Note: TLI, TRAP and RESET events can interrupt a level 3 program.

Interrupt software priority registers (ISPRX)

Read/Write (bit 7:4 of **ISPR3** are read only)

Reset Value: 1111 1111 (FFh)

Table 10. Dedicated interrupt instruction set (continued)

Instruction	New description	Function/Example	I1	H	I0	N	Z	C
TRAP	Software trap	Software NMI	1		1			
WFI	Wait for interrupt		1		0			

Note: During the execution of an interrupt routine, the HALT, POPCC, RIM, SIM and WFI instructions change the current software priority up to the next IRET instruction or one of the previously mentioned instructions.
In order not to lose the current software priority level, the RIM, SIM, HALT, WFI and POP CC instructions should never be used in an interrupt routine.

Table 11. Interrupt mapping

N°	Source block	Description	Register label	Priority order	Exit from HALT	Address vector
	RESET	Reset	N/A	Highest Priority ↓ Lowest Priority	yes	FFFEh-FFFFh
	TRAP	Software Interrupt			no	FFFCh-FFFDh
0	ICP	FLASH Start programming NMI interrupt (TLI)				FFFAh-FFFBh
1	UART	ISO7816-3 UART Interrupt	UIC			FFF8h-FFF9h
2	USB	USB Communication Interrupt	USBISTR			FFF6h-FFF7h
3	WAKUP1	External Interrupt Port C			yes	FFF4h-FFF5h
4	WAKUP2	External Interrupt Port A			yes	FFF2h-FFF3h
5	TIM	TBU Timer Interrupt	TBUSR		no	FFF0h-FFF1h
6	CARDDET [1)]	Smartcard Insertion/Removal Interrupt [1)]	USCUR		yes	FFEEh-FFEFh
7	ESUSP	End suspend Interrupt	USBISTR			FFECh-FFEDh
8	Not used				no	FFEAh-FFEBh

Note: This interrupt can be used to exit from USB suspend mode.

8 Power saving modes

8.1 Introduction

To give a large measure of flexibility to the application in terms of power consumption, two main power saving modes are implemented in the ST7.

After a RESET the normal operating mode is selected by default (RUN mode). This mode drives the device (CPU and embedded peripherals) by means of a master clock which is based on the main oscillator frequency.

From Run mode, the different power saving modes may be selected by setting the relevant register bits or by calling the specific ST7 software instruction whose action depends on the oscillator status.

8.2 Wait mode

WAIT mode places the MCU in a low power consumption mode by stopping the CPU.

This power saving mode is selected by calling the “WFI” ST7 software instruction.

All peripherals remain active. During WAIT mode, the I bit of the CC register is forced to 0, to enable all interrupts. All other registers and memory remain unchanged. The MCU remains in WAIT mode until an interrupt or Reset occurs, whereupon the Program Counter branches to the starting address of the interrupt or Reset service routine.
The MCU will remain in WAIT mode until a Reset or an Interrupt occurs, causing it to wake up.

Refer to *Figure 21*.

12 On-chip peripherals

12.1 Watchdog timer (WDG)

12.1.1 Introduction

The Watchdog timer is used to detect the occurrence of a software fault, usually generated by external interference or by unforeseen logical conditions, which causes the application program to abandon its normal sequence. The Watchdog circuit generates an MCU reset on expiry of a programmed time period, unless the program refreshes the counter's contents before the T6 bit becomes cleared.

12.1.2 Main features

- Programmable free-running downcounter (64 increments of 65536 CPU cycles)
- Programmable reset
- Reset (if watchdog activated) when the T6 bit reaches zero
- Hardware Watchdog selectable by option byte
- Watchdog Reset indicated by status flag

12.1.3 Functional description

The counter value stored in the CR register (bits T[6:0]), is decremented every 65,536 machine cycles, and the length of the timeout period can be programmed by the user in 64 increments.

If the watchdog is activated (the WDGA bit is set) and when the 7-bit timer (bits T[6:0]) rolls over from 40h to 3Fh (T6 becomes cleared), it initiates a reset cycle pulling low the reset pin for typically 500ns.

The application program must write in the CR register at regular intervals during normal operation to prevent an MCU reset. This downcounter is free-running: it counts down even if the watchdog is disabled. The value to be stored in the CR register must be between FFh and C0h (see *Table 18*).

- The WDGA bit is set (watchdog enabled)
- The T6 bit is set to prevent generating an immediate reset
- The T[5:0] bits contain the number of increments which represents the time delay before the watchdog produces a reset.

Table 18. Watchdog timing (f_{CPU} = 8 MHz)

	CR register initial value	WDG timeout period (ms)
Max	FFh	524.288
Min	C0h	8.192

Error status register (ERRSR)

Read only

Reset Value: 0000 0000 (00h)

7							0
0	0	0	0	0	ERR2	ERR1	ERR0

Bits 7:3 = Reserved, forced by hardware to 0.

Bits 2:0 = **ERR[2:0]** *Error type*.

These bits identify the type of error which occurred.

ERR2	ERR1	ERR0	Meaning
0	0	0	No error
0	0	1	Bitstuffing error
0	1	0	CRC error
0	1	1	EOP error (unexpected end of packet or SE0 not followed by J-state)
1	0	0	PID error (PID encoding error, unexpected or unknown PID)
1	0	1	Memory over / underrun (memory controller has not answered in time to a memory data request)
1	1	1	Other error (wrong packet, timeout error)

Note: These bits are set by hardware when an error interrupt occurs and are reset automatically when the error bit (USBISTR bit 4) is cleared by software.

Endpoint 0 register (EP0R)

Read/Write

Reset value: 0000 0000(00h)

7							0
CTR0	DTOG_TX	STAT_ TX1	STAT_ TX0	0	DTOG_RX	STAT_ RX1	STAT_ RX0

This register is used for controlling Endpoint 0.
Bits 6:4 and bits 2:0 are also reset by a USB reset, either received from the USB or forced through the FRES bit in USBCTLR.

Bit 7 = **CTR0** *Correct Transfer*.
This bit is set by hardware when a correct transfer operation is performed on Endpoint 0.
This bit must be cleared after the corresponding interrupt has been serviced.

0: No CTR on Endpoint 0

1: Correct transfer on Endpoint 0

Table 20. Reception status encoding

STAT_RX1	STAT_RX0	Meaning
1	0	**NAK**: the endpoint is NAKed and all reception requests result in a NAK handshake.
1	1	**VALID**: this endpoint is enabled (if an address match occurs, the USB interface handles the transaction).

These bits are written by software. Hardware sets the STAT_RX and STAT_TX bits to NAK when a correct transfer has occurred (CTR=1) addressed to this endpoint, so the software has the time to examine the received data before acknowledging a new transaction.

Note: If a SETUP transaction is received while the status is different from DISABLED, it is acknowledged and the two directional status bits are set to NAK by hardware.

When a STALL is answered by the USB device, the two directional status bits are set to STALL by hardware.

Endpoint transmission register (EP1TXR, EP2TXR, EP3TXR, EP4TXR, EP5TXR)

Read/Write

Reset value: 0000 0000 (00h)

7							0
0	0	0	0	CTR_TX	DTOG_TX	STAT_TX1	STAT_TX0

This register is used for controlling Endpoint 1, 2, 3, 4 or 5 transmission. Bits 2:0 are also reset by a USB reset, either received from the USB or forced through the FRES bit in the USBCTLR register.

Bits [7:4] = Reserved, forced by hardware to 0.

Bit 3 = **CTR_TX** *Correct Transmission Transfer*.
This bit is set by hardware when a correct transfer operation is performed in transmission. This bit must be cleared after the corresponding interrupt has been serviced.

0: No CTR in transmission on Endpoint 1, 2, 3, 4 or 5

1: Correct transfer in transmission on Endpoint 1, 2, 3, 4 or 5

Bit 2 = **DTOG_TX** *Data Toggle, for transmission transfers*.
This bit contains the required value of the toggle bit (0=DATA0, 1=DATA1) for the next data packet. DTOG_TX toggles only when the transmitter has received the ACK signal from the USB host. DTOG_TX and DTOG_RX are normally updated by hardware, at the receipt of a relevant PID. They can be also written by the user, both for testing purposes and to force a specific (DATA0 or DATA1) token.

Bits [1:0] = **STAT_TX [1:0]** *Status bits, for transmission transfers*.
These bits contain the information about the endpoint status, which is listed below

At the end of a reception, the value of this register is the max size decremented by the number of bytes received (to determine the number of bytes received, the software must subtract the content of this register from the allocated buffer size).

Transmission counter register (CNT2TXR)

Read/Write

Reset Value 0000 0000 (00h)

7							0
0	CNT6	CNT5	CNT4	CNT3	CNT2	CNT1	CNT0

This register contains the number of bytes to be transmitted by Endpoint 2 at the next IN token addressed to it.

Table 23. USB register map and reset values

Address (Hex.)	Register name	7	6	5	4	3	2	1	0
20	USBISTR Reset Value	CTR 0	0 0	SOVR 0	ERR 0	SUSP 0	ESUSP 0	RESET 0	SOF 0
21	USBIMR Reset Value	CTRM 0	0 0	SOVRM 0	ERRM 0	SUSPM 0	ESUSPM 0	RESETM 0	SOFM 0
22	USBCTLR Reset Value	RSM 0	USB_RST 0	0	0	RESUME 0	PDWN 1	FSUSP 1	FRES 0
23	DADDR Reset Value	0	ADD6 0	ADD5 0	ADD4 0	ADD3 0	ADD2 0	ADD1 0	ADD0 0
24	USBSR Reset Value	PID1 0	PID0 0	IN /OUT 0	0	0	EP2 0	EP1 0	EP0 0
25	EP0R Reset Value	CTR0 0	DTOG_TX 0	STAT_TX 1 0	STAT_TX 0 0	0 0	DTOG_RX 0	STAT_RX 1 0	STAT_RX 0 0
26	CNT0RXR Reset Value	0	0	0	0	CNT3 0	CNT2 0	CNT1 0	CNT0 0
27	CNT0TXR Reset Value	0	0	0	0	CNT3 0	CNT2 0	CNT1 0	CNT0 0
28	EP1TXR Reset Value	0	0	0	0	CTR_TX 0	DTOG_TX 0	STAT_TX 1 0	STAT_TX 0 0
29	CNT1TXR Reset Value	0	0	0	0	CNT3 0	CNT2 0	CNT1 0	CNT0 0
2A	EP2RXR Reset Value	0	0	0	0	CTR_RX 0	DTOG_RX 0	STAT_RX 1 0	STAT_RX 0 0
2B	CNT2RXR Reset Value	0	CNT6 0	CNT5 0	CNT4 0	CNT3 0	CNT2 0	CNT1 0	CNT0 0

Bit 7 =**TxBEF** *Transmit Buffer Empty Flag.*

- Read only

0: Transmit buffer is not empty

1: Transmit buffer is empty

Bit 6 = **CRDIRF** *Card Insertion/Removal Flag.*
- Read only

0: No card is present

1: A card is present

Bit 5 = **IOVF** *Card Overload Current Flag.*
- Read only

0: No card overload current

1: Card overload current

Bit 4 = **VCARDOK** *Card voltage status Flag.*

- Read only

0: The card voltage is not in the specified range

1: The card voltage is within the specified range

Bit 3 = **WTF** *Waiting Time Counter overflow Flag.*
- Read only

0: The WT Counter has not reached its maximum value

1: The WT Counter has reached its maximum value

Bit 2 = **TXCF** *Transmitted character Flag.*
- Read/Write
This bit is set by hardware and cleared by software.

0: No character transmitted

1: A character has been transmitted

Bit 1 = **RXCF** *Received character Flag.*
- Read only
This bit is set by hardware and cleared by hardware when the CRDRXB buffer is read.

0: No character received

1: A character has been received

1: Compensation mode enabled. To allow non integer value, one clock cycle is subtracted from the ETU value on odd bits. See *Figure 32*.

Bit [6:3] = Reserved

Bits 2:0 = **ETU [10:8]** *ETU value in card clock cycles.*

Writing CRDETU1 register reloads the ETU counter.

CRDETU0
Read/Write
Reset Value: 0111 0100 (74h)

7							0
ETU7	ETU6	ETU5	ETU4	ETU3	ETU2	ETU1	ETU0

Bits 7:0 = **ETU [7:0]** *ETU value in card clock cycles.*

Note: The value of ETU [10:0] must in the range 12 to 2047. To write 2048, clear all the bits.

Guardtime register (CRDGTx)

CRDGT1
Read/Write
Reset Value: 0000 0000 (00h)

7							0
0	0	0	0	0	0	0	GT8

CRDGT0
Read/Write
Reset Value: 0000 1100 (0Ch)

7							0
GT7	GT6	GT5	GT4	GT3	GT2	GT1	GT0

Software writes the Guardtime value in this register. The value is loaded at the end of the current Guard period.

GT: Guard Time: Minimum time between two consecutive start bits in transmission mode. Value expressed in Elementary Time Units (from 11 to 511).

The Guardtime between the last byte received from the card and the next byte transmitted by the reader must be handled by software.

0: No RXC interrupt pending

1: RXC interrupt pending

Bit 0 = **PARP** *Parity Error interrupt pending.*

This bit is set by hardware when a PAR event occurs and the PARM bit is set.

0: No PAR interrupt pending

1: PAR interrupt pending

Smartcard transmit buffer (CRDTXB)
Read/Write
Reset Value: 0000 0000 (00h)

7							0
TB7	TB6	TB5	TB4	TB3	TB2	TB1	TB0

This register is used to send a byte to the smartcard.

Smartcard receive buffer (CRDRXB)
Read
Reset Value: 0000 0000 (00h)

7							0
RB7	RB6	RB5	RB4	RB3	RB2	RB1	RB0

This register is used to receive a byte from the smartcard.

Table 24. Register map and reset values

Address (Hex.)	Register label	7	6	5	4	3	2	1	0
00	**CRDCR** Reset Value	CRDRST 0	DETCNF 0	VCARD1 0	VCARD0 0	UART 0	WTEN 0	CREP 0	CONV 0
01	**CRDSR** Reset Value	TXBEF 1	CRDIRF 0	IOVF 0	VCARDOK 0	WTF 0	TXCF 0	RXCF 0	PARF 0
02	**CRDCCR** Reset Value	CLKSEL 0	- 0	CRDC8 x	CRDC4 x	CRDIO x	CRDCLK 0	CRDRST x	CRDVCC 0
03	**CRDETU1** Reset Value	COMP 0	- 0	- 0	- 0	- 0	ETU10 1	ETU9 0	ETU8 0
04	**CRDETU0** Reset Value	ETU7 0	ETU6 1	ETU5 1	ETU4 1	ETU3 0	ETU2 1	ETU1 0	ETU0 0

Table 24. Register map and reset values (continued)

Address (Hex.)	Register label	7	6	5	4	3	2	1	0
05	**CRDGT1** Reset Value	- 0	- 0	- 0	- 0	- 0	- 0	- 0	GT8 0
06	**CRDGT0** Reset Value	GT7 0	GT6 0	GT5 0	GT4 0	GT3 1	GT2 1	GT1 0	GT0 0
07	**CRDWT2** Reset Value	WT23 0	WT22 0	WT21 0	WT20 0	WT19 0	WT18 0	WT17 0	WT16 0
08	**CRDWT1** Reset Value	WT15 0	WT14 0	WT13 1	WT12 0	WT11 0	WT10 1	WT9 0	WT8 1
09	**CRDWT0** Reset Value	WT7 1	WT6 0	WT5 0	WT4 0	WT3 0	WT2 0	WT1 0	WT0 0
0A	**CRDIER** Reset Value	TXBEM 0	- 0	IOVM 0	VCRDM 0	WTM 0	TXCM 0	RXCM 0	PARM 0
0B	**CRDIPR** Reset Value	TXBEP 0	- 0	IOVP 0	VCRDP	WTP 0	TXCP 0	RXCP 0	PARP 0
0C	**CRDTXB** Reset Value	TB7 0	TB6 0	TB5 0	TB4 0	TB3 0	TB2 0	TB1 0	TB0 0
0D	**CRDRXB** Reset Value	RB7 0	RB6 0	RB5 0	RB4 0	RB3 0	RB2 0	RB1 0	RB0 0

13 Instruction set

13.1 CPU addressing modes

The CPU features 17 different addressing modes which can be classified in 7 main groups:

Addressing mode	Example
Inherent	nop
Immediate	ld A,#$55
Direct	ld A,$55
Indexed	ld A,($55,X)
Indirect	ld A,([$55],X)
Relative	jrne loop
Bit operation	bset byte,#5

The CPU Instruction set is designed to minimize the number of bytes required per instruction: To do so, most of the addressing modes may be subdivided in two sub-modes called long and short:

- Long addressing mode is more powerful because it can use the full 64-Kbyte address space, however it uses more bytes and more CPU cycles.
- Short addressing mode is less powerful because it can generally only access page zero (0000h - 00FFh range), but the instruction size is more compact, and faster. All memory to memory instructions use short addressing modes only (CLR, CPL, NEG, BSET, BRES, BTJT, BTJF, INC, DEC, RLC, RRC, SLL, SRL, SRA, SWAP)

The ST7 Assembler optimizes the use of long and short addressing modes.

Table 25. CPU addressing mode overview

Mode			Syntax	Destination	Pointer address	Pointer size (Hex.)	Length (bytes)
Inherent			nop				+ 0
Immediate			ld A,#$55				+ 1
Short	Direct		ld A,$10	00..FF			+ 1
Long	Direct		ld A,$1000	0000..FFFF			+ 2
No Offset	Direct	Indexed	ld A,(X)	00..FF			+ 0
Short	Direct	Indexed	ld A,($10,X)	00..1FE			+ 1
Long	Direct	Indexed	ld A,($1000,X)	0000..FFFF			+ 2
Short	Indirect		ld A,[$10]	00..FF	00..FF	byte	+ 2
Long	Indirect		ld A,[$10.w]	0000..FFFF	00..FF	word	+ 2
Short	Indirect	Indexed	ld A,([$10],X)	00..1FE	00..FF	byte	+ 2
Long	Indirect	Indexed	ld A,([$10.w],X)	0000..FFFF	00..FF	word	+ 2
Relative	Direct		jrne loop	PC+/-127			+ 1

Table 25. CPU addressing mode overview (continued)

Mode			Syntax	Destination	Pointer address	Pointer size (Hex.)	Length (bytes)
Relative	Indirect		jrne [$10]	PC+/-127	00..FF	byte	+ 2
Bit	Direct		bset $10,#7	00..FF			+ 1
Bit	Indirect		bset [$10],#7	00..FF	00..FF	byte	+ 2
Bit	Direct	Relative	btjt $10,#7,skip	00..FF			+ 2
Bit	Indirect	Relative	btjt [$10],#7,skip	00..FF	00..FF	byte	+ 3

13.1.1 Inherent

All Inherent instructions consist of a single byte. The opcode fully specifies all the required information for the CPU to process the operation.

Inherent instruction	Function
NOP	No operation
TRAP	S/W Interrupt
WFI	Wait For Interrupt (Low Power Mode)
HALT	Halt Oscillator (Lowest Power Mode)
RET	Sub-routine Return
IRET	Interrupt Sub-routine Return
SIM	Set Interrupt Mask (level 3)
RIM	Reset Interrupt Mask (level 0)
SCF	Set Carry Flag
RCF	Reset Carry Flag
RSP	Reset Stack Pointer
LD	Load
CLR	Clear
PUSH/POP	Push/Pop to/from the stack
INC/DEC	Increment/Decrement
TNZ	Test Negative or Zero
CPL, NEG	1 or 2 Complement
MUL	Byte Multiplication
SLL, SRL, SRA, RLC, RRC	Shift and Rotate Operations
SWAP	Swap Nibbles

14.7.2 Electro magnetic interference (EMI)

Based on a simple application running on the product (toggling 2 LEDs through the I/O ports), the product is monitored in terms of emission. This emission test is in line with the norm SAE J 1752/3 which specifies the board and the loading of each pin.

Symbol	Parameter	Conditions	Monitored frequency band	Max vs. [f_{OSC}/f_{CPU}]		Unit
				4/8MHz	4/4MHz	
S_{EMI}	Peak level	V_{DD}=5V, T_A=+25°C, conforming to SAE J 1752/3	0.1MHz to 30MHz	19	18	dBµV
			30MHz to 130MHz	32	27	
			130MHz to 1GHz	31	26	
			SAE EMI Level	4	3.5	-

Note: Data based on characterization results, not tested in production.

14.7.3 Absolute maximum ratings (electrical sensitivity)

Based on three different tests (ESD, LU and DLU) using specific measurement methods, the product is stressed in order to determine its performance in terms of electrical sensitivity. For more details, refer to the application note AN1181.

Electro-static discharge (ESD)

Electro-Static Discharges (a positive then a negative pulse separated by 1 second) are applied to the pins of each sample according to each pin combination. The sample size depends on the number of supply pins in the device (3 parts*(n+1) supply pin). The Human Body Model is simulated. This test conforms to the JESD22-A114A standard.

Table 36. Absolute maximum ratings

Symbol	Ratings	Conditions	Maximum value (1)	Unit
$V_{ESD(HBM)}$	Electro-static discharge voltage (Human Body Model)	T_A=+25°C	2000	V

1. Data based on characterization results, not tested in production.

Static and dynamic latch-up

- **LU**: 3 complementary static tests are required on 10 parts to assess the latch-up performance. A supply overvoltage (applied to each power supply pin) and a current injection (applied to each input, output and configurable I/O pin) are performed on each sample. This test conforms to the EIA/JESD 78 IC latch-up standard. For more details, refer to the application note AN1181.
- **DLU**: Electro-Static Discharges (one positive then one negative test) are applied to each pin of 3 samples when the micro is running to assess the latch-up performance in

16.3 ST7 Application notes

Table 42. ST7 Application notes

Identification	Description
Application Examples	
AN1658	Serial Numbering Implementation
AN1720	Managing the Read-out Protection in Flash Microcontrollers
AN1755	A High Resolution/precision Thermometer Using ST7 and NE555
AN1756	Choosing a DALI Implementation Strategy with ST7DALI
AN1812	A High Precision, Low Cost, Single Supply ADC for Positive and Negative Input Voltages
Example Drivers	
AN 969	SCI Communication Between ST7 and PC
AN 970	SPI Communication Between ST7 and EEPROM
AN 971	I²C Communication Between ST7 and M24Cxx EEPROM
AN 972	ST7 Software SPI Master Communication
AN 973	SCI Software Communication with a PC Using ST72251 16-Bit Timer
AN 974	Real Time Clock with ST7 Timer Output Compare
AN 976	Driving a Buzzer Through ST7 Timer PWM Function
AN 979	Driving an Analog Keyboard with the ST7 ADC
AN 980	ST7 Keypad Decoding Techniques, Implementing Wake-Up on Keystroke
AN1017	Using the ST7 Universal Serial Bus Microcontroller
AN1041	Using ST7 PWM Signal to Generate Analog Output (Sinusoïd)
AN1042	ST7 Routine for I²C Slave Mode Management
AN1044	Multiple Interrupt Sources Management for ST7 MCUs
AN1045	ST7 S/W Implementation of I²C Bus Master
AN1046	UART Emulation Software
AN1047	Managing Reception Errors with the ST7 SCI Peripherals
AN1048	ST7 Software LCD Driver
AN1078	PWM Duty Cycle Switch Implementing True 0% & 100% Duty Cycle
AN1082	Description of the ST72141 Motor Control Peripherals Registers
AN1083	ST72141 BLDC Motor Control Software and Flowchart Example
AN1105	ST7 pCAN Peripheral Driver
AN1129	PWM Management for BLDC Motor Drives Using the ST72141
AN1130	An Introduction to Sensorless Brushless DC Motor Drive Applications with the ST72141
AN1148	Using the ST7263 for Designing a USB Mouse
AN1149	Handling Suspend Mode on a USB Mouse
AN1180	Using the ST7263 Kit to Implement a USB Game Pad

Table 42. ST7 Application notes (continued)

Identification	Description
AN1796	Field Updates for FLASH Based ST7 Applications Using a PC Comm Port
AN1900	Hardware Implementation for ST7DALI-EVAL
AN1904	ST7MC Three-phase AC Induction Motor Control Software Library
AN1905	ST7MC Three-phase BLDC Motor Control Software Library
System Optimization	
AN1711	Software Techniques for Compensating ST7 ADC Errors
AN1827	Implementation of SIGMA-DELTA ADC with ST7FLITE05/09
AN2009	PWM Management for 3-Phase BLDC Motor Drives Using the ST7FMC
AN2030	Back EMF Detection During PWM On Time by ST7MC

16.4 Important notes

16.4.1 Unexpected reset fetch

If an interrupt request occurs while a "POP CC" instruction is executed, the interrupt controller does not recognise the source of the interrupt and, by default, passes the RESET vector address to the CPU.

Workaround

To solve this issue, a "POP CC" instruction must always be preceded by a "SIM" instruction.

16.4.2 Flash devices only

The behavior described in the following section (*Section 16.4.3*) is present on Rev W ST7FSCR devices only.

They are identifiable:

- on the device package, by the last letter of the Trace Code marked on the device package.
- on the box, by the last 3 digits of the Internal Sales Type printed in the box label.

Table 43. Device identification

	Trace code marked on device	Internal sales type on box label
Flash Devices:	“xxxxxxxxW”	7FSCR1R4T1$U6 7FSCR1E4M1$U6

See also *Figure 48*.