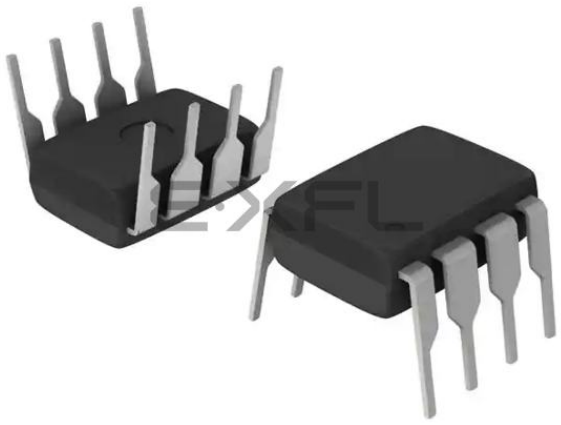


Welcome to E-XFL.COM

What is "Embedded - Microcontrollers"?

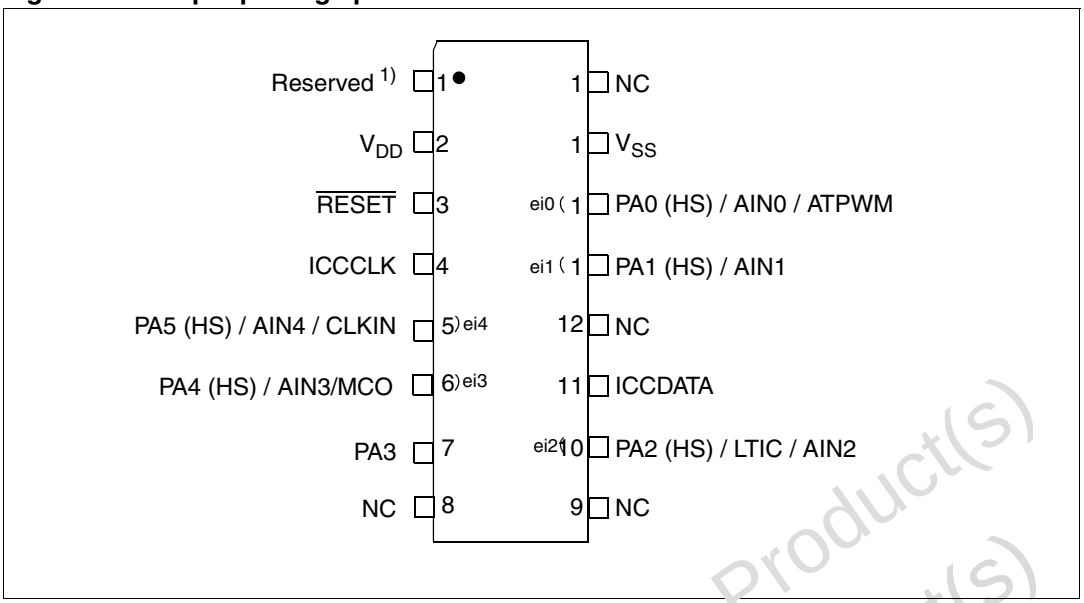
"Embedded - Microcontrollers" refer to small, integrated circuits designed to perform specific tasks within larger systems. These microcontrollers are essentially compact computers on a single chip, containing a processor core, memory, and programmable input/output peripherals. They are called "embedded" 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	-
Peripherals	LVD, POR, PWM, WDT
Number of I/O	5
Program Memory Size	1KB (1K x 8)
Program Memory Type	FLASH
EEPROM Size	-
RAM Size	128 x 8
Voltage - Supply (Vcc/Vdd)	2.4V ~ 5.5V
Data Converters	-
Oscillator Type	Internal
Operating Temperature	-40°C ~ 85°C (TA)
Mounting Type	Through Hole
Package / Case	8-DIP (0.300", 7.62mm)
Supplier Device Package	-
Purchase URL	https://www.e-xfl.com/product-detail/stmicroelectronics/st7fliteus2b6

Contents

Figure 4. 16-pin package pinout



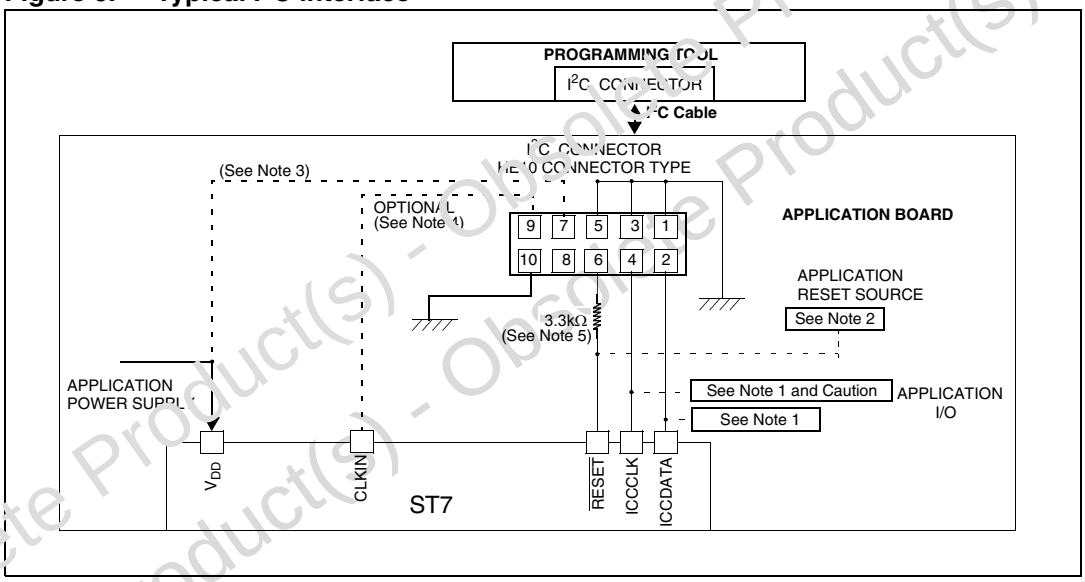
1. Reserved pins must be tied to ground.
2. The differences versus the 8-pin packages are listed below:
   The I^2C signals (ICCCLK and ICCDATA) are mapped on dedicated pins.
   The RESET signal is mapped on a dedicated pin. It is not multiplexed with PA3.
   PA3 pin is always configured as output. Any change on multiplexed IO reset control registers (MUXCR1 and MUXCR2) will have no effect on PA3 functionality. Refer to *Section 6.5: Register description on page 37*.

Pin 9 has to be connected to the CLKIN pin of the ST7 when I^2C mode is selected with option bytes disabled (35-pulse I^2C entry mode). When option bytes are enabled (38-pulse I^2C entry mode), the internal RC clock (internal RC or AWU RC) is forced. If internal RC is selected in the option byte, the internal RC is provided. If AWU RC or external clock is selected, the AWU RC oscillator is provided.

A serial resistor must be connected to I^2C connector pin 6 in order to prevent contention on PA3/$\overline{\text{RESET}}$ pin. Contention may occur if a tool forces a state on $\overline{\text{RESET}}$ pin while PA3 pin forces the opposite state in output mode. The resistor value is defined to limit the current below 2 mA at 5 V. If PA3 is used as output push-pull, then the application must be switched off to allow the tool to take control of the $\overline{\text{RESET}}$ pin (PA3). To allow the programming tool to drive the $\overline{\text{RESET}}$ pin below V_{IL}, special care must also be taken when a pull-up is placed on PA3 for application reasons.

Caution: During normal operation, ICCCLK pin must be pulled- up, internally or externally (external pull-up of 10 kΩ mandatory in noisy environment). This is to avoid entering I^2C mode unexpectedly during a reset. In the application, even if the pin is configured as output, any reset will put it back in input pull-up.

Figure 6. Typical I^2C interface



4.5 Memory protection

There are two different types of memory protection: readout protection and Write/Erase Protection which can be applied individually.

4.5.1 Readout protection

Readout protection, when selected provides a protection against program memory content extraction and against write access to Flash memory. Even if no protection can be considered as totally unbreakable, the feature provides a very high level of protection for a general purpose microcontroller. Program memory is protected.

5 Central processing unit

5.1 Introduction

This CPU has a full 8-bit architecture and contains six internal registers allowing efficient 8-bit data manipulation.

5.2 Main features

- 63 basic instructions
- Fast 8-bit by 8-bit multiply
- 17 main addressing modes
- Two 8-bit index registers
- 16-bit stack pointer
- Low power modes
- Maskable hardware interrupts
- Non-maskable software interrupt

5.3 CPU registers

The six CPU registers shown in *Figure 7* are not present in the memory mapping and are accessed by specific instructions.

5.3.1 Accumulator (A)

The Accumulator is an 8-bit general purpose register used to hold operands and the results of the arithmetic and logic calculations and to manipulate data.

5.3.2 Index registers (X and Y)

In indexed addressing modes, these 8-bit registers are used to create either effective addresses or temporary storage areas for data manipulation. (The cross-assembler generates a precede instruction (PRE) to indicate that the following instruction refers to the Y register.)

The Y register is not affected by the interrupt automatic procedures (not pushed to and popped from the stack).

5.3.3 Program counter (PC)

The program counter is a 16-bit register containing the address of the next instruction to be executed by the CPU. It is made of two 8-bit registers PCL (program counter low which is the LSB) and PCH (program counter high which is the MSB).

Bit 7:5 Set to ‘1’

Bit 4 **H** *Half carry*

This bit is set by hardware when a carry occurs between bits 3 and 4 of the ALU during an ADD or ADC instruction. It is reset by hardware during the same instructions.

0: No half carry has occurred.

1: A half carry has occurred.

This bit is tested using the JRH or JRNH instruction. The H bit is useful in BCD arithmetic subroutines.

Bit 3 **I** *Interrupt mask*

This bit is set by hardware when entering in interrupt or by software to disable all interrupts except the TRAP software interrupt. This bit is cleared by software.

0: Interrupts are enabled.

1: Interrupts are disabled.

This bit is controlled by the RIM, SIM and IRET instructions and is tested by the JRM and JRNM instructions.

Note: Interrupts requested while I is set are latched and can be processed when I is cleared. By default an interrupt routine is not interruptible because the I bit is set by hardware at the start of the routine and reset by the IRET instruction at the end of the routine. If the I bit is cleared by software in the interrupt routine, pending interrupts are serviced regardless of the priority level of the current interrupt routine.

Bit 2 **N** *Negative*

This bit is set and cleared by hardware. It is representative of the result sign of the last arithmetic, logical or data manipulation. It is a copy of the 7^{th} bit of the result.

0: The result of the last operation is positive or null.

1: The result of the last operation is negative
(that is, the most significant bit is a logic 1).

This bit is accessed by the JRMI and JRPL instructions.

Bit 1 **Z** *Zero*

This bit is set and cleared by hardware. This bit indicates that the result of the last arithmetic, logical or data manipulation is zero.

0: The result of the last operation is different from zero.

1: The result of the last operation is zero.

This bit is accessed by the JREQ and JRNE test instructions.

Bit 0 = **C** *Carry/borrow*

This bit is set and cleared by hardware and software. It indicates an overflow or an underflow has occurred during the last arithmetic operation.

0: No overflow or underflow has occurred.

1: An overflow or underflow has occurred.

This bit is driven by the SCF and RCF instructions and tested by the JRC and JRNC instructions. It is also affected by the “bit test and branch”, shift and rotate instructions.

This 16-bit register is read/write by software but can be written only once between two reset events. It is cleared by hardware after a reset; When both MUXCR0 and MUXCR1 registers are at 00h, the multiplexed PA3/$\overline{\text{RESET}}$ pin will act as $\overline{\text{RESET}}$. To configure this pin as output (Port A3), write 55h to MUXCR0 and AAh to MUXCR1.

These registers are one-time writable only.

- To configure PA3 as general purpose output:
  After power-on / reset, the application program has to configure the I/O port by writing to these registers as described above. Once the pin is configured as an I/O output, it cannot be changed back to a reset pin by the application code.
- To configure PA3 as $\overline{\text{RESET}}$:
  An internally generated reset (such as POR, LVD, WDG, illegal opcode) will clear the two registers and the pin will act again as a reset function. Otherwise, a power-down is required to put the pin back in reset configuration.

Table 8. Multiplexed IO register map and reset values

Address (Hex.)	Register label	7	6	5	4	3	2	1	0
0047h	MUXCR0 reset value	MIR7 0	MIR6 0	MIR5 0	MIR4 0	MIR3 0	MIR2 0	MIR1 0	MIR0 0
0048h	MUXCR1 reset value	MIR15 0	MIR14 0	MIR13 0	MIR12 0	MIR11 0	MIR10 0	MIR9 0	MIR8 0

7 Interrupts

The ST7 core may be interrupted by one of two different methods: Maskable hardware interrupts as listed in the “interrupt mapping” table and a non-maskable software interrupt (TRAP). The Interrupt processing flowchart is shown in *Figure 14*.

The maskable interrupts must be enabled by clearing the I bit in order to be serviced. However, disabled interrupts may be latched and processed when they are enabled (see external interrupts subsection).

Note: After reset, all interrupts are disabled.

When an interrupt has to be serviced:

- Normal processing is suspended at the end of the current instruction execution.
- The PC, X, A and CC registers are saved onto the stack.
- The I bit of the CC register is set to prevent additional interrupts.
- The PC is then loaded with the interrupt vector of the interrupt to service and the first instruction of the interrupt service routine is fetched (refer to the Interrupt Mapping table for vector addresses).

The interrupt service routine should finish with the IRET instruction which causes the contents of the saved registers to be recovered from the stack.

Note: As a consequence of the IRET instruction, the I bit is cleared and the main program resumes.

Priority management

By default, a servicing interrupt cannot be interrupted because the I bit is set by hardware entering in interrupt routine.

In the case when several interrupts are simultaneously pending, an hardware priority defines which one will be serviced first (see *Table 9: Interrupt mapping*).

Interrupts and low power mode

All interrupts allow the processor to leave the Wait low power mode. Only external and specifically mentioned interrupts allow the processor to leave the Halt low power mode (refer to the “Exit from Halt” column in *Table 9: Interrupt mapping*).

7.1 Non maskable software interrupt

This interrupt is entered when the TRAP instruction is executed regardless of the state of the I bit. It is serviced according to the flowchart in *Figure 14*.

Halt mode recommendations

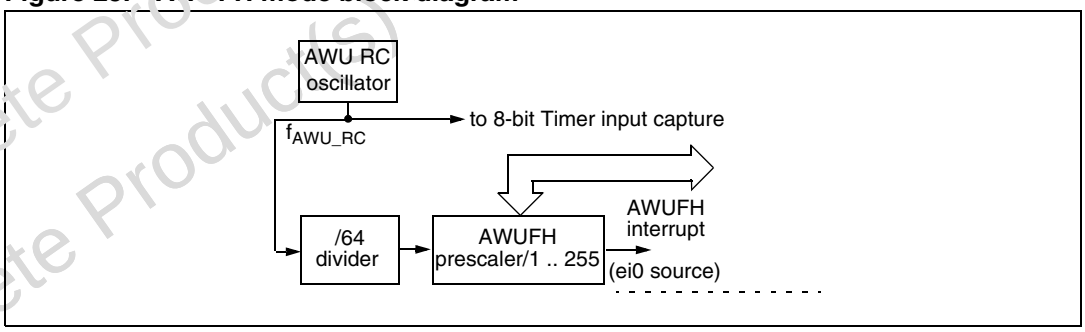
- Make sure that an external event is available to wakeup the microcontroller from Halt mode.
- When using an external interrupt to wakeup the microcontroller, reinitialize the corresponding I/O as “Input Pull-up with Interrupt” before executing the HALT instruction. The main reason for this is that the I/O may be wrongly configured due to external interference or by an unforeseen logical condition.
- For the same reason, reinitialize the level sensitiveness of each external interrupt as a precautionary measure.
- The opcode for the HALT instruction is 0x8E. To avoid an unexpected HALT instruction due to a program counter failure, it is advised to clear all occurrences of the data value 0x8E from memory. For example, avoid defining a constant in ROM with the value 0x8E.
- As the HALT instruction clears the I bit in the CC register to allow interrupts, the user may choose to clear all pending interrupt bits before executing the HALT instruction. This avoids entering other peripheral interrupt routines after executing the external interrupt routine corresponding to the wakeup event (reset or external interrupt).

8.5 Auto-wakeup from Halt mode

Auto-wakeup from Halt (AWUFH) mode is similar to Halt mode with the addition of a specific internal RC oscillator for wakeup (Auto-wakeup from Halt oscillator) which replaces the main clock which was active before entering Halt mode. Compared to Active-halt mode, AWUFH has lower power consumption (the main clock is not kept running), but there is no accurate realtime clock available.

It is entered by executing the HALT instruction when the AWUEN bit in the AWUCSR register has been set.

Figure 25. AWUFH mode block diagram



As soon as Halt mode is entered, and if the AWUEN bit has been set in the AWUCSR register, the AWU RC oscillator provides a clock signal (f_{AWU_RC}). Its frequency is divided by a fixed divider and a programmable prescaler controlled by the AWUPR register. The output of this prescaler provides the delay time. When the delay has elapsed, the following actions are performed:

- The AWUF flag is set by hardware,
- An interrupt wakes-up the MCU from Halt mode,
- The main oscillator is immediately turned on and the 64 CPU cycle delay is used to stabilize it.

After this startup delay, the CPU resumes operation by servicing the AWUFH interrupt. The AWU flag and its associated interrupt are cleared by software reading the AWUCSR register.

To compensate for any frequency dispersion of the AWU RC oscillator, it can be calibrated by measuring the clock frequency f_{AWU_RC} and then calculating the right prescaler value. Measurement mode is enabled by setting the AWUM bit in the AWUCSR register in Run mode. This connects f_{AWU_RC} to the input capture of the 8-bit lite timer, allowing the f_{AWU_RC} to be measured using the main oscillator clock as a reference timebase.

Similarities with Halt mode

The following AWUFH mode behavior is the same as normal Halt mode:

- The MCU can exit AWUFH mode by means of any interrupt with exit from Halt capability or a reset (see *Section 8.4: Active-halt and Halt modes*).
- When entering AWUFH mode, the I bit in the CC register is forced to 0 to enable interrupts. Therefore, if an interrupt is pending, the MCU wakes up immediately.
- In AWUFH mode, the main oscillator is turned off causing all internal processing to be stopped, including the operation of the on-chip peripherals. None of the peripherals are clocked except those which get their clock supply from another clock generator (such as an external or auxiliary oscillator like the AWU oscillator).
- The compatibility of watchdog operation with AWUFH mode is configured by the WDGHALT option bit in the option byte. Depending on this setting, the HALT instruction when executed while the watchdog system is enabled, can generate a watchdog reset.

Figure 26. AWUF Halt timing diagram

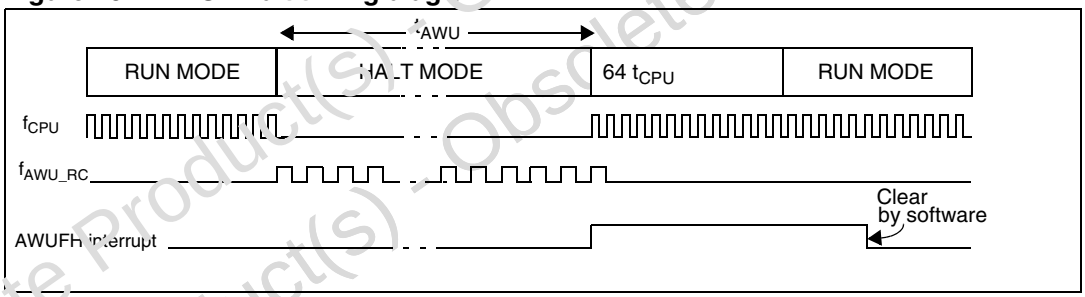
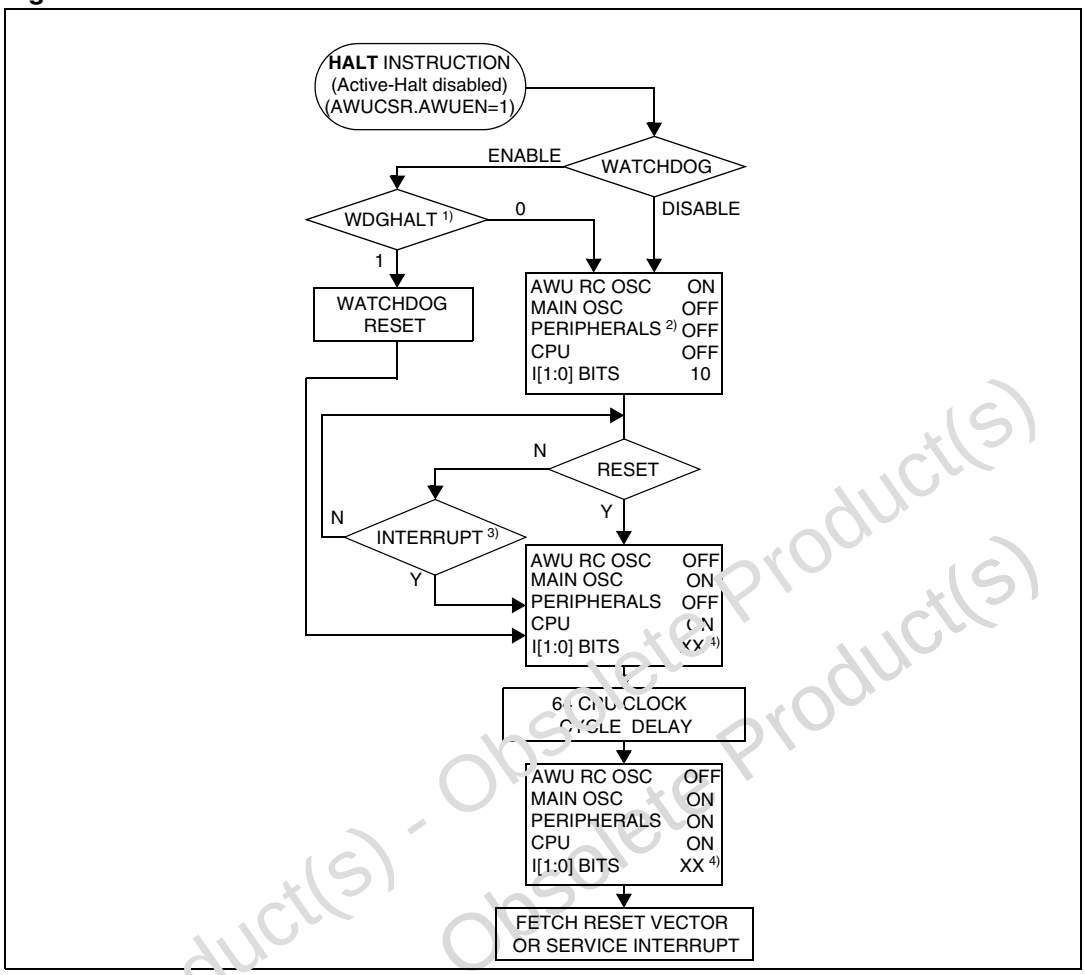


Figure 27. AWUFH mode flowchart



1. WDGHALT is an option bit. See option byte section for more details.
2. Peripheral clocked with an external clock source can still be active.
3. Only an AWUFH interrupt and some specific interrupts can exit the MCU from Halt mode (such as external interrupt). Refer to *Table 9: Interrupt mapping* for more details.
4. Before servicing an interrupt, the CC register is pushed on the stack. The I[1:0] bits of the CC register are set to the current software priority level of the interrupt routine and recovered when the CC register is popped.

Spurious interrupts

When enabling/disabling an external interrupt by setting/resetting the related OR register bit, a spurious interrupt is generated if the pin level is low and its edge sensitivity includes falling/rising edge. This is due to the edge detector input which is switched to '1' when the external interrupt is disabled by the OR register.

To avoid this unwanted interrupt, a "safe" edge sensitivity (rising edge for enabling and falling edge for disabling) has to be selected before changing the OR register bit and configuring the appropriate sensitivity again.

Caution: In case a pin level change occurs during these operations (asynchronous signal input), as interrupts are generated according to the current sensitivity, it is advised to disable all interrupts before and to reenable them after the complete previous sequence in order to avoid an external interrupt occurring on the unwanted edge.

This corresponds to the following steps:

1. To enable an external interrupt:
   a) Set the interrupt mask with the SIM instruction (in cases where a pin level change could occur)
   b) Select rising edge
   c) Enable the external interrupt through the OR register
   d) Select the desired sensitivity if different from rising edge
   e) Reset the interrupt mask with the RIM instruction (in cases where a pin level change could occur)
2. To disable an external interrupt:
   a) Set the interrupt mask with the SIM instruction SIM (in cases where a pin level change could occur)
   b) Select falling edge
   c) Disable the external interrupt through the OR register
   d) Select rising edge

9.2.2 Output modes

The output configuration is selected by setting the corresponding DDR register bit. In this case, writing the DR register applies this digital value to the I/O pin through the latch. Then reading the DR register returns the previously stored value.

Two different output modes can be selected by software through the OR register: Output push-pull and open-drain.

Table 17. DR register value and output pin status(1)

DR	Push-pull	Open-drain
0	V_{SS}	V_{SS}
1	V_{DD}	Floating

1. When switching from input to output mode, the DR register has to be written first to drive the correct level on the pin as soon as the port is configured as an output.

ADC data register high (ADCDRH)

Reset value: 0000 0000 (00h)

7							0
D9	D8	D7	D6	D5	D4	D3	D2
Read only							

Bits 7:0 D[9:2] *MSB of Analog Converted value*

ADC control/data register Low (ADCDRL)

Reset value: 0000 0000 (00h)

7							0
0	0	0	0	SLOW	0	D1	D0
Read/write							

Bits 7:4 Reserved. Forced by hardware to 0.

Bit 3 **SLOW** *Slow mode*
This bit is set and cleared by software. It is used together with the SPEED bit to configure the ADC clock speed as shown on the table below (see *Table 33: Configuring the ADC clock speed*).

Bit 2 Reserved. Forced by hardware to 0.

Bits 1:0 **D[1:0]** *LSB of Analog Converted value*

Table 33. Configuring the ADC clock speed

f_{ADC}	SLOW	SPEED
$f_{CPU}/2$	0	0
f_{CPU}	0	1
$f_{CPU}/4$	1	x

Table 34. ADC register map and reset values

Address (Hex.)	Register label	7	6	5	4	3	2	1	0
0034h	**ADCCSR** Reset value	EOC 0	SPEED 0	ADON 0	0 0	0 0	CH2 0	CH1 0	CH0 0
0035h	**ADCDRH** Reset value	D9 0	D8 0	D7 0	D6 0	D5 0	D4 0	D3 0	D2 0
0036h	**ADCDRL** Reset value	0 0	0 0	0 0	0 0	SLOW 0	0 0	D1 0	D0 0

Table 42. Illegal opcode detection (continued)

Mnemo	Description	Function/example	Dst	Src	H	I	N	Z	C
NEG	Negate (2's compl)	neg $10	reg, M	-	-	-	N	Z	C
NOP	No operation		-	-	-	-	-	-	-
OR	OR operation	A = A + M	A	M	-	-	N	Z	-
POP	Pop from the stack	pop reg	reg	M	-	-	-	-	-
		pop CC	CC	M	H	I	N	Z	C
PUSH	Push onto the stack	push Y	M	reg, CC	-	-	-	-	-
RCF	Reset carry flag	C = 0	-	-	-	-	-	-	0
RET	Subroutine return		-	-	-	-	-	-	-
RIM	Enable Interrupts	I = 0	-	-	-	0	-	-	-
RLC	Rotate left true C	C ≤ Dst ≤ C	reg, M	-	-	-	N	Z	C
RRC	Rotate right true C	C ≥ Dst ≥ C	reg, M	-	-	-	N	Z	C
RSP	Reset Stack Pointer	S = Max allowed	-	-	-	-	-	-	-
SBC	Subtract with carry	A = A - M - C	A	M	-	-	N	Z	C
SCF	Set carry flag	C = 1	-	-	-	-	-	-	1
SIM	Disable interrupts	I = 1	-	-	-	1	-	-	-
SLA	Shift left arithmetic	C ≤ Dst ≤ 0	reg, M	-	-	-	N	Z	C
SLL	Shift left logic	C ≤ Dst ≤ 0	reg, M	-	-	-	N	Z	C
SRL	Shift right logic	0 ≥ Dst ≥ C	reg, M	-	-	-	0	Z	C
SRA	Shift right arithmetic	Dst7 ≥ Dst ≥ C	reg, M	-	-	-	N	Z	C
SUB	Subtraction	A = A - M	A	M	-	-	N	Z	C
SWAP	SWAP nibbles	Dst[7..4] ≤ ≥Dst[3..0]	reg, M	-	-	-	N	Z	-
TNZ	Test for Neg & Zero	tnz lbl1	-	-	-	-	N	Z	-
TRAP	S/W trap	S/W interrupt	-	-	-	1		-	-
WFI	Wait for interrupt		-	-	-	0	-	-	-
XOR	Exclusive OR	A = A XOR M	A	M	-	-	N	Z	-

13.2 Thermal characteristics

Table 74. Thermal characteristics

Symbol	Ratings		Value	Unit
R_{thJA}	Package thermal resistance (junction to ambient)	Plastic DIP8	82	°C/W
		SO8	130	
		DFN8 (on 4-layer PCB) DFN8 (on 2-layer PCB)	50 106	
T_{Jmax}	Maximum junction temperature(1)		150	°C
P_{Dmax}	Power dissipation(2)	Plastic DIP8	300	mW
		SO8	180	
		DFN8 (on 4-layer PCB)	500	
		DFN8 (on 2-layer PCB)	250	

1. The maximum chip-junction temperature is based on technology characteristics.
2. The maximum power dissipation is obtained from the formula $P_D = (T_J - T_A) / R_{thJA}$.
The power dissipation of an application can be defined by the user with the formula: $P_D = P_{INT} + P_{PORT}$ where P_{INT} is the chip internal power ($I_{DD} \times V_{DD}$) and P_{PORT} is the port power dissipation depending on the ports used in the application.

14.3 Development tools

Development tools for the ST7 microcontrollers include a complete range of hardware systems and software tools from STMicroelectronics and third-party tool suppliers. The range of tools includes solutions to help you evaluate microcontroller peripherals, develop and debug your application, and program your microcontrollers.

14.3.1 Starter kits

ST offers complete, affordable **starter kits**. Starter kits are complete, affordable hardware/software tool packages that include features and samples to help you quickly start developing your application.

14.3.2 Development and debugging tools

Application development for ST7 is supported by fully optimizing **C Compilers** and the **ST7 Assembler-Linker** toolchain, which are all seamlessly integrated in the ST7 integrated development environments in order to facilitate the debugging and fine-tuning of your application. The Cosmic C Compiler is available in a free version that outputs up to 16 Kbytes of code.

The range of hardware tools includes full-featured **ST7-EMU3 series emulators,** cost effective **ST7-DVP3 series emulators** and the low-cost **RLink** in-circuit debugger/programmer. These tools are supported by the **ST7 Toolset** from STMicroelectronics, which includes the ST7VD7 integrated development environment (IDE) with high-level language debugger, editor, project manager and integrated programming interface.

14.3.3 Programming tools

During the development cycle, the **ST7-DVP3** and **ST7-EMU3 series emulators** and the **RLink** provide in-circuit programming capability for programming the Flash microcontroller on your application board.

ST also provides a low-cost dedicated in-circuit programmer, the **ST7-STICK**, as well as **ST7 Socket Boards** which provide all the sockets required for programming any of the devices in a specific ST7 sub-family on a platform that can be used with any tool with in-circuit programming capability for ST7.

For production programming of ST7 devices, ST's third-party tool partners also provide a complete range of gang and automated programming solutions, which are ready to integrate into your production environment.

14.3.4 Order codes for development and programming tools

Table 80 below lists the ordering codes for the ST7LITEUSx development and programming tools. For additional ordering codes for spare parts and accessories, refer to the online product selector at www.st.com/mcu.

Table 81. ST7 application notes (continued)

Identification	Description
AN1077	Overview of enhanced CAN controllers for ST7 and ST9 MCUs
AN1086	U435 can-do solutions for car multiplexing
AN1103	Improved B-EMF detection for low speed, low voltage with ST72141
AN1150	Benchmark ST72 vs PC16
AN1151	Performance comparison between ST72254 & PC16F876
AN1278	LIN (local interconnect network) solutions
Product migration	
AN1131	Migrating applications from ST72511/311/214/124 to ST72521/321/324
AN1322	Migrating an application from ST7263 Rev.B to ST7263B
AN1365	Guidelines for migrating ST72C254 applications to ST72F264
AN1604	How to use ST7MDT1-TRAIN with ST72F264
AN2200	Guidelines for migrating ST7LITE1x applications to ST7FLITE1xB
Product optimization	
AN 982	Using ST7 with ceramic resonator
AN1014	How to Minimize the ST7 power consumption
AN1015	Software techniques for improving microcontroller EMC performance
AN1040	Monitoring the Vbus signal for USB Self-powered devices
AN1070	ST7 checksum self-checking capability
AN1181	Electrostatic Discharge sensitive measurement
AN1324	Calibrating the RC oscillator of the ST7FLITE0 MCU using the mains
AN1502	Emulated data EEPROM with ST7 HDFLASH memory
AN1529	Extending the current & voltage capability on the ST7265 VDDF supply
AN1530	Accurate timebase for low-cost ST7 applications with internal RC oscillator
AN1605	Using an active RC to wakeup the ST7LITE0 from power saving mode
AN1636	Understanding and minimizing ADC conversion errors
AN1828	PIR (passive Infrared) detector using the ST7FLITE05/09/SUPERLITE
AN1946	Sensorless BLDC motor control and BEMF sampling methods with ST7MC
AN1953	PFC for ST7MC starter kit
AN1971	ST7LITE0 microcontrolled ballast
Programming and tools	
AN 978	ST7 Visual Develop software key debugging features
AN 983	Key features of the Cosmic ST7 C-compiler package
AN 985	Executing code In ST7 RAM
AN 986	Using the indirect addressing mode with ST7

15 Known limitations

External interrupt 2 (ei2)

Whatever the external interrupt sensitivity configured through EICR1 register, ei2 cannot exit the MCU from Halt, Active-halt and AWUFH modes when a falling edge occurs.

Workaround

None

Table 82. Document revision history (continued)

Date	Revision	Changes
18-Sep-06	3	Modified description of AVD[1:0] bits in the AVDTRH register in *Section 7.4.4* Modified description of CNTR[11:0] bits in *Section 10.2.6: Register description* Modified values in *Table 44* LVD and AVD tables updated, *Table 47*, *Table 48* and *Table 49* Internal RC oscillator data modified in *Table 50* and new table added *Table 51* Typical data in *Table 54* (on chip peripherals) modified EMC characteristics updated, *Section 12.7* R_{PU} data corrected in *Table 63* including additional notes Output driving current table updated, *Table 64* R_{ON} data corrected in *Table 65*. Modified ADC accuracy tables in *Section 12.10* *Section :* updated Errata sheet removed from document Notes modified for low voltage detector *Section 7.4.1* Notes updated in *Section 4.4* (I^2C Interface) Thermal characteristics table updated, *Table 74* Modified option list on *Section 14.2: Ordering information* Modified *Section 14.3: Development tools* Modified text in *Section :*
26-Jan-07	4	Added -40°C to 125°C temperature range Modified note on ei4 in *Table 9. Interrupt mapping* Added note 3 to *Section 7.4.2: External Interrupt Control register 2 (EICR2)* Added *Figure 41* and *Figure 40* Added a note to LVDRF in *Section 7.4.4: Register description* *Section 6.4.1: Introduction* Modified *Table 47* and *Table 48* Modified *Table 50*Updated *Table 53* Updated *Table 64* Modified R_{AIN} and ADC accuracy tables in *Section 12.10: ADC characteristics* Modified *Table 80* Modified *Table 79* Modified option list on *Figure 69: Option list*
06-Feb-2009	5	Document reformatted. Replaced ST7ULTRALITE by ST7LITEUS2 and ST7LITEUS5. Removed limitations in user and in I^2C mode from *Section 15: Known limitations*, and added *External interrupt 2 (ei2)*. Added MCO on pin 3. Updated *Section 12.3.2: Operating conditions with low voltage detector (LVD)*, *Section 12.3.3: Auxiliary voltage detector (AVD) thresholds*, *Section 12.3.4: Internal RC oscillator*, *Section 12.4: Supply current characteristics*, and *Section 12.8.2: Output driving current characteristics*. Updated internal RC prescaler to add 500 KHz. Updated ECOPACK text in *Section 13.1: Package mechanical data*. Added PDIP16 silhouette on cover page, and updated *Table 73: 16-pin plastic dual in-line package, 300-mil width, package mechanical data* and *Figure 68: 16-pin plastic dual in-line package, 300-mil width, package outline*. Changed order codes to die A version in *Table 79: Supported order codes*. Removed soldering information section. Updated option list.