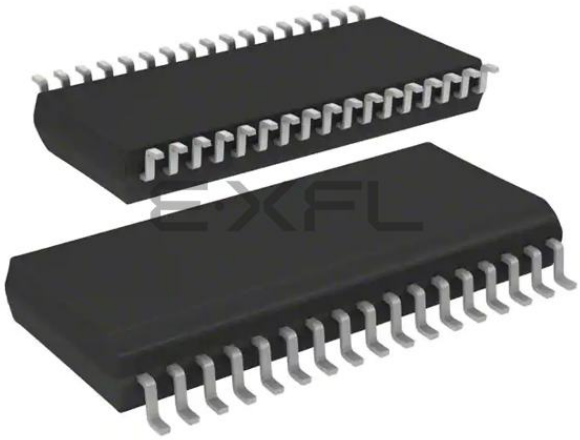


Welcome to E-XFL.COM

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	I²C, SCI, USB
Peripherals	DMA, LVD, POR, PWM, WDT
Number of I/O	19
Program Memory Size	32KB (32K x 8)
Program Memory Type	FLASH
EEPROM Size	-
RAM Size	384 x 8
Voltage - Supply (Vcc/Vdd)	4V ~ 5.5V
Data Converters	A/D 8x8b
Oscillator Type	Internal
Operating Temperature	0°C ~ 70°C (TA)
Mounting Type	Surface Mount
Package / Case	34-BSOP (0.295", 7.50mm Width)
Supplier Device Package	-
Purchase URL	https://www.e-xfl.com/product-detail/stmicroelectronics/st72f63bk6m1

13.11 8-bit ADC 162

14 Package characteristics 165

14.1 Package mechanical data 166

14.2 Thermal characteristics 171

14.3 Soldering and glueability information 171

15 Device configuration and ordering information 172

15.1 Option byte 172

15.2 Device ordering information and transfer of customer code 173

15.3 Development tools 174

15.3.1 Evaluation tools and starter kits 174

15.3.2 Development and debugging tools 174

15.3.3 Programming tools 175

15.3.4 Order codes for ST7263Bx development tools 175

15.4 ST7 application notes 177

16 Known limitations 181

16.1 PA2 limitation with OCMP1 enabled 181

16.2 Unexpected RESET fetch 181

16.3 USB behavior with LVD disabled 181

16.4 I2C multimaster 181

16.5 Halt mode power consumption with ADC on 182

16.6 SCI wrong BREAK duration 182

17 Revision history 184

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 7th 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.

Stack Pointer (SP)

Reset value: 017Fh

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	1	0	SP6	SP5	SP4	SP3	SP2	SP1	SP0
Read/write															

The Stack Pointer is a 16-bit register which is always pointing to the next free location in the stack. It is then decremented after data has been pushed onto the stack and incremented before data is popped from the stack (see *Figure 10*).

Since the stack is 128 bytes deep, the 9 most significant bits are forced by hardware. Following an MCU Reset, or after a Reset Stack Pointer instruction (RSP), the Stack Pointer contains its reset value (the SP6 to SP0 bits are set) which is the stack higher address.

The least significant byte of the Stack Pointer (called S) can be directly accessed by a LD instruction.

Note: When the lower limit is exceeded, the Stack Pointer wraps around to the stack upper limit, without indicating the stack overflow. The previously stored information is then overwritten and therefore lost. The stack also wraps in case of an underflow.

The stack is used to save the return address during a subroutine call and the CPU context during an interrupt. The user may also directly manipulate the stack by means of the PUSH and POP instructions. In the case of an interrupt, the PCL is stored at the first location

6 Reset and clock management

6.1 Reset

The Reset procedure is used to provide an orderly software start-up or to exit low power modes.

Three reset modes are provided: a low voltage (LVD) reset, a watchdog reset and an external reset at the $\overline{\text{RESET}}$ pin.

A reset causes the reset vector to be fetched from addresses FFFEh and FFFFh in order to be loaded into the PC and with program execution starting from this point.

An internal circuitry provides a 4096 CPU clock cycle delay from the time that the oscillator becomes active.

Caution: When the ST7 is unprogrammed or fully erased, the Flash is blank and the RESET vector is not programmed. For this reason, it is recommended to keep the RESET pin in low state until programming mode is entered, in order to avoid unwanted behavior.

6.1.1 Low voltage detector (LVD)

Low voltage reset circuitry generates a reset when V_{DD} is:

- Below V_{IT+} when V_{DD} is rising
- Below V_{IT-} when V_{DD} is falling

During low voltage reset, the $\overline{\text{RESET}}$ pin is held low, thus permitting the MCU to reset other devices.

It is recommended to make sure that the V_{DD} supply voltage rises monotonously when the device is exiting from Reset, to ensure the application functions properly.

6.1.2 Watchdog reset

When a watchdog reset occurs, the $\overline{\text{RESET}}$ pin is pulled low permitting the MCU to reset other devices in the same way as the low voltage reset (*Figure 12*).

6.1.3 External reset

The external reset is an active low input signal applied to the $\overline{\text{RESET}}$ pin of the MCU. As shown in *Figure 15*, the $\overline{\text{RESET}}$ signal must stay low for a minimum of one and a half CPU clock cycles.

An internal Schmitt trigger at the $\overline{\text{RESET}}$ pin is provided to improve noise immunity.

9 I/O ports

9.1 Introduction

The I/O ports offer different functional modes:

- Transfer of data through digital inputs and outputs and for specific pins
- Analog signal input (ADC)
- Alternate signal input/output for the on-chip peripherals
- External interrupt generation

An I/O port consists of up to 8 pins. Each pin can be programmed independently as a digital input (with or without interrupt generation) or a digital output.

9.2 Functional description

Each port is associated to 2 main registers:

- Data register (DR)
- Data Direction register (DDR)

Each I/O pin may be programmed using the corresponding register bits in DDR register: bit X corresponding to pin X of the port. The same correspondence is used for the DR register.

Table 10. I/O pin functions

DDR	Mode
0	Input
1	Output

Input modes

The input configuration is selected by clearing the corresponding DDR register bit.

In this case, reading the DR register returns the digital value applied to the external I/O pin.

Note: 1 All the inputs are triggered by a Schmitt trigger.

2 When switching from input mode to output mode, the DR register should be written first to output the correct value as soon as the port is configured as an output.

Interrupt function

When an I/O is configured as an input with interrupt, an event on this I/O can generate an external interrupt request to the CPU. The interrupt sensitivity is given independently according to the description mentioned in the ITRFRE interrupt register.

Each pin can independently generate an interrupt request.

Each external interrupt vector is linked to a dedicated group of I/O port pins (see interrupts section). If more than one input pin is selected simultaneously as an interrupt source, this is logically ORed. For this reason if one of the interrupt pins is tied low, the other ones are masked.

9.3.6 Related documentation

AN1045: S/W implementation of I^2C bus master

AN1048: Software LCD driver

11.2 16-bit timer

11.2.1 Introduction

The timer consists of a 16-bit free-running counter driven by a programmable prescaler.

It may be used for a variety of purposes, including pulse length measurement of up to two input signals (*input capture*) or generation of up to two output waveforms (*output compare* and *PWM*).

Pulse lengths and waveform periods can be modulated from a few microseconds to several milliseconds using the timer prescaler and the CPU clock prescaler.

Some ST7 devices have two on-chip 16-bit timers. They are completely independent, and do not share any resources. They are synchronized after a MCU reset as long as the timer clock frequencies are not modified.

This description covers one or two 16-bit timers. In ST7 devices with two timers, register names are prefixed with TA (Timer A) or TB (Timer B).

11.2.2 Main features

- Programmable prescaler: f_{CPU} divided by 2, 4 or 8
- Overflow status flag and maskable interrupt
- External clock input (must be at least four times slower than the CPU clock speed) with the choice of active edge
- 1 or 2 output Compare functions each with:
  - 2 dedicated 16-bit registers
  - 2 dedicated programmable signals
  - 2 dedicated status flags
  - 1 dedicated maskable interrupt
- 1 or 2 input Capture functions each with:
  - 2 dedicated 16-bit registers
  - 2 dedicated active edge selection signals
  - 2 dedicated status flags
  - 1 dedicated maskable interrupt
- Pulse width modulation mode (PWM)
- One Pulse mode
- Reduced Power mode
- 5 alternate functions on I/O ports (ICAP1, ICAP2, OCMP1, OCMP2, EXTCLK)*

The Block Diagram is shown in *Figure 27*.

Note: Some timer pins may not be available (not bonded) in some ST7 devices. Refer to the device pin out description.
When reading an input signal on a non-bonded pin, the value will always be '1'.

One Pulse mode

One Pulse mode enables the generation of a pulse when an external event occurs. This mode is selected via the OPM bit in the CR2 register.

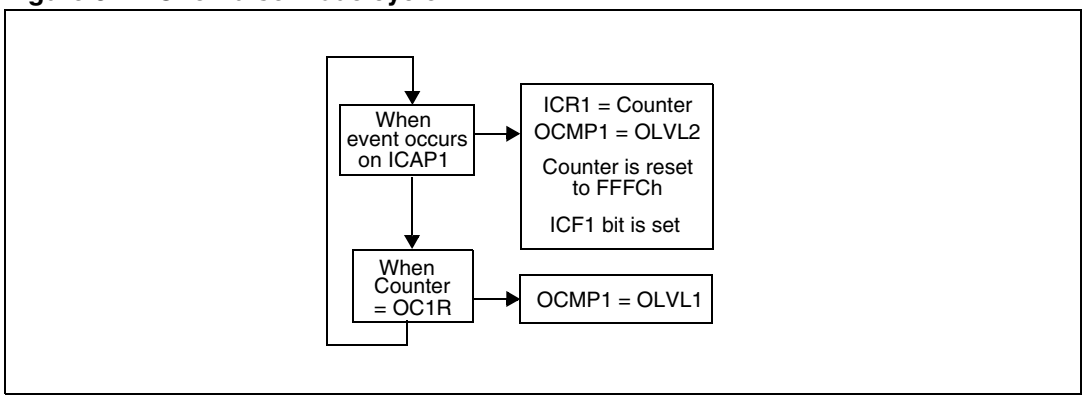
The One Pulse mode uses the input Capture1 function and the output Compare1 function.

Procedure

To use One Pulse mode:

1. Load the OC1R register with the value corresponding to the length of the pulse (see the formula in the opposite column).
2. Select the following in the CR1 register:
   - Using the OLVL1 bit, select the level to be applied to the OCMP1 pin after the pulse.
   - Using the OLVL2 bit, select the level to be applied to the OCMP1 pin during the pulse.
   - Select the edge of the active transition on the ICAP1 pin with the IEDG1 bit (the ICAP1 pin must be configured as floating input).
3. Select the following in the CR2 register:
   - Set the OC1E bit, the OCMP1 pin is then dedicated to the output Compare 1 function.
   - Set the OPM bit.
   - Select the timer clock CC[1:0] (see *Table 24*).

Figure 37. One Pulse mode cycle

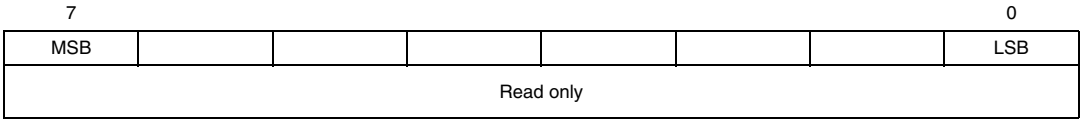


Then, on a valid event on the ICAP1 pin, the counter is initialized to FFFCh and OLVL2 bit is loaded on the OCMP1 pin, the ICF1 bit is set and the value FFFDh is loaded in the IC1R register.

Because the ICF1 bit is set when an active edge occurs, an interrupt can be generated if the ICIE bit is set.

Clearing the input Capture interrupt request (that is, clearing the ICF*i* bit) is done in two steps:

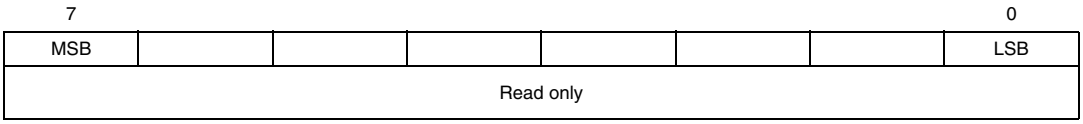
1. Reading the SR register while the ICF*i* bit is set.
2. An access (read or write) to the IC*i*LR register.



Alternate Counter Low register (ACLR)

Reset value: 1111 1100 (FCh)

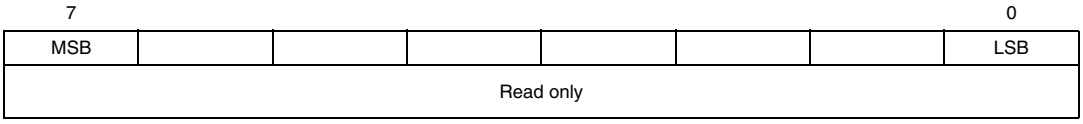
This is an 8-bit register that contains the low part of the counter value. A write to this register resets the counter. An access to this register after an access to CSR register does not clear the TOF bit in the CSR register.



Input Capture 2 High register (IC2HR)

Reset value: Undefined

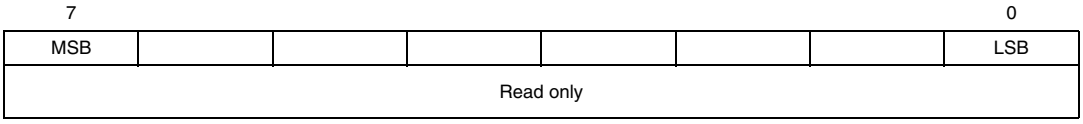
This is an 8-bit read only register that contains the high part of the counter value (transferred by the input Capture 2 event).



Input Capture 2 Low register (IC2LR)

Reset value: Undefined

This is an 8-bit read only register that contains the low part of the counter value (transferred by the input Capture 2 event).



Transmitter

The transmitter can send data words of either 8 or 9 bits depending on the M bit status. When the M bit is set, word length is 9 bits and the 9th bit (the MSB) has to be stored in the T8 bit in the SCICR1 register.

Character Transmission

During an SCI transmission, data shifts out least significant bit first on the TDO pin. In this mode, the SCIDR register consists of a buffer (TDR) between the internal bus and the transmit shift register (see *Figure 41*).

Procedure

1. Select the M bit to define the word length.
2. Select the desired baud rate using the SCIBRR and the SCIETPR registers.
3. Set the TE bit to assign the TDO pin to the alternate function and to send a idle frame as first transmission.
4. Access the SCISR register and write the data to send in the SCIDR register (this sequence clears the TDRE bit). Repeat this sequence for each data to be transmitted.

Clearing the TDRE bit is always performed by the following software sequence:

1. An access to the SCISR register
2. A write to the SCIDR register

The TDRE bit is set by hardware and it indicates:

- The TDR register is empty.
- The data transfer is beginning.
- The next data can be written in the SCIDR register without overwriting the previous data.

This flag generates an interrupt if the TIE bit is set and the I bit is cleared in the CC register.

When a transmission is taking place, a write instruction to the SCIDR register stores the data in the TDR register and which is copied in the shift register at the end of the current transmission.

When no transmission is taking place, a write instruction to the SCIDR register places the data directly in the shift register, the data transmission starts, and the TDRE bit is immediately set.

When a frame transmission is complete (after the stop bit or after the break frame) the TC bit is set and an interrupt is generated if the TCIE is set and the I bit is cleared in the CC register.

Clearing the TC bit is performed by the following software sequence:

1. An access to the SCISR register
2. A write to the SCIDR register

Note: The TDRE and TC bits are cleared by the same software sequence.

Break characters

Setting the SBK bit loads the shift register with a break character. The break frame length depends on the M bit (see *Figure 42*).

PID register (PIDR)

Reset value: xx00 0000 (x0h)

7							0
TP3	TP2	0	0	0	RX_ SEZ	RXD	0
Read only							

[7:6] **TP[3:2]** *Token PID bits 3 & 2.*
USB token PIDs are encoded in four bits. **TP[3:2]** correspond to the variable token PID bits 3 & 2.
: PID bits 1 & 0 have a fixed value of 01.

Note: When a CTR interrupt occurs (see register ISTR) the software should read the TP3 and TP2 bits to retrieve the PID name of the token received. The USB standard defines TP bits (see Table 33).

[5:3] Reserved. Forced by hardware to 0.

2 **RX_SEZ** *Received single-ended zero*
This bit indicates the status of the RX_SEZ transceiver output.
0: No SE0 (single-ended zero) state
1: USB lines are in SE0 (single-ended zero) state

1 **RXD** *Received data*
0: No K-state
1: USB lines are in K-state
This bit indicates the status of the RXD transceiver output (differential receiver output).
If the environment is noisy, the RX_SEZ and RXD bits can be used to secure the application. By interpreting the status, software can distinguish a valid End Suspend event from a spurious wakeup due to noise on the external USB line. A valid End Suspend is followed by a Resume or Reset sequence. A Resume is indicated by RXD=1, a Reset is indicated by RX_SEZ=1.

0 Reserved. Forced by hardware to 0.

Table 33. TP bit definition

TP3	TP2	PID Name
0	0	OUT
1	0	IN
1	1	SETUP

2 **ESUSP** *End suspend mode.*
This bit is set by hardware when, during suspend mode, activity is detected that wakes the USB interface up from suspend mode.
This interrupt is serviced by a specific vector, in order to wake up the ST7 from Halt mode.
0: No End Suspend detected
1: End Suspend detected

1 **RESET** *USB reset.*
This bit is set by hardware when the USB reset sequence is detected on the bus.
0: No USB reset signal detected
1: USB reset signal detected
Note: The DADDR, EP0RA, EP0RB, EP1RA, EP1RB, EP2RA and EP2RB registers are reset by a USB reset.

0 **SOF** *Start of frame.*
This bit is set by hardware when a low-speed SOF indication (keep-alive strobe) is seen on the USB bus. It is also issued at the end of a resume sequence.
0: No SOF signal detected
1: SOF signal detected

Note: To avoid spurious clearing of some bits, it is recommended to clear them using a load instruction where all bits which must not be altered are set, and all bits to be cleared are reset. Avoid read-modify-write instructions like AND, XOR.

Interrupt Mask register (IMR)

These bits are mask bits for all interrupt condition bits included in the ISTR. Whenever one of the IMR bits is set, if the corresponding ISTR bit is set, and the I bit in the CC register is cleared, an interrupt request is generated. For an explanation of each bit, please refer to the corresponding bit description in ISTR.

Reset value: 0000 0000 (00h)

7							0
SUSPM	DOVRM	CTRM	ERRM	IOVRM	ESUSPM	RESETM	SOFM
Read.write							

Note: Once transmission and/or reception are enabled, registers EPnRA and/or EPnRB (respectively) must not be modified by software, as the hardware can change their value on the fly.

When the operation is completed, they can be accessed again to enable a new operation.

Interrupt handling

Start of Frame (SOF)

The interrupt service routine may monitor the SOF events for a 1 ms synchronization event to the USB bus. This interrupt is generated at the end of a resume sequence and can also be used to detect this event.

USB Reset (RESET)

When this event occurs, the DADDR register is reset, and communication is disabled in all endpoint registers (the USB interface will not respond to any packet). Software is responsible for reenabling endpoint 0 within 10 ms of the end of reset. To do this, set the STAT_RX bits in the EP0RB register to VALID.

Suspend (SUSP)

The CPU is warned about the lack of bus activity for more than 3 ms, which is a suspend request. The software should set the USB interface to suspend mode and execute an ST7 HALT instruction to meet the USB-specified power constraints.

End Suspend (ESUSP)

The CPU is alerted by activity on the USB, which causes an ESUSP interrupt. The ST7 automatically terminates Halt mode.

Correct Transfer (CTR)

1. When this event occurs, the hardware automatically sets the STAT_TX or STAT_RX to NAK. Every valid endpoint is NAKed until software clears the CTR bit in the ISTR register, independently of the endpoint number addressed by the transfer which generated the CTR interrupt. If the event triggering the CTR interrupt is a SETUP transaction, both STAT_TX and STAT_RX are set to NAK.
2. Read the PIDR to obtain the token and the IDR to get the endpoint number related to the last transfer. When a CTR interrupt occurs, the TP3-TP2 bits in the PIDR register and EP1-EP0 bits in the IDR register stay unchanged until the CTR bit in the ISTR register is cleared.
3. Clear the CTR bit in the ISTR register.

Table 36. USB register map and reset values

Address (Hex.)	Register Name	7	6	5	4	3	2	1	0
25	PIDR Reset value	TP3 x	TP2 x	0 0	0 0	0 0	RX_SEZ 0	RXD 0	0 0
26	DMAR Reset value	DA15 x	DA14 x	DA13 x	DA12 x	DA11 x	DA10 x	DA9 x	DA8 x

Following the address reception and after SR1 register has been read, the slave receives bytes from the SDA line into the DR register via the internal shift register. After each byte the interface generates in sequence:

- Acknowledge pulse if the ACK bit is set
- EVF and BTF bits are set with an interrupt if the ITE bit is set.

Then the interface waits for a read of the SR1 register followed by a read of the DR register, **holding the SCL line low** (see *Figure 48* Transfer sequencing EV2).

Slave transmitter

Following the address reception and after the SR1 register has been read, the slave sends bytes from the DR register to the SDA line via the internal shift register.

The slave waits for a read of the SR1 register followed by a write in the DR register, **holding the SCL line low** (see *Figure 48* Transfer sequencing EV3).

When the acknowledge pulse is received, the EVF and BTF bits are set by hardware with an interrupt if the ITE bit is set.

Closing Slave communication

After the last data byte is transferred a Stop Condition is generated by the master. The interface detects this condition and sets EVF and STOPF bits with an interrupt if the ITE bit is set.

Then the interface waits for a read of the SR2 register (see *Figure 48* Transfer sequencing EV4).

Error cases

- **BERR**: Detection of a Stop or a Start condition during a byte transfer. In this case, the EVF and the BERR bits are set with an interrupt if the ITE bit is set.
  If it is a Stop, then the interface discards the data, released the lines and waits for another Start condition.
  If it is a Start, then the interface discards the data and waits for the next slave address on the bus.
- **AF**: Detection of a non-acknowledge bit. In this case, the EVF and AF bits are set with an interrupt if the ITE bit is set.
  The AF bit is cleared by reading the I2CSR2 register. However, if read before the completion of the transmission, the AF flag will be set again, thus possibly generating a new interrupt. Software must ensure either that the SCL line is back at 0 before reading the SR2 register, or be able to correctly handle a second interrupt during the 9th pulse of a transmitted byte.

Note: In case of errors, SCL line is not held low; however, the SDA line can remain low if the last bits transmitted are all 0. While AF=1, the SCL line may be held low due to SB or BTF flags that are set at the same time. It is then necessary to release both lines by software.

How to Release the SDA / SCL lines

Set and subsequently clear the STOP bit while BTF is set. The SDA/SCL lines are released after the transfer of the current byte.

Master mode

To switch from default Slave mode to Master mode, a Start condition generation is needed.

Start condition

Table 49. Inherent instructions (continued)

Inherent instruction	Function
CPL, NEG	1 or 2 Complement
MUL	Byte Multiplication
SLL, SRL, SRA, RLC, RRC	Shift and Rotate Operations
SWAP	Swap Nibbles

12.1.2 Immediate instructions

Immediate instructions have two bytes, the first byte contains the opcode, the second byte contains the operand value.

Table 50. Immediate instructions

Immediate instruction	Function
LD	Load
CP	Compare
BCP	Bit Compare
AND, OR, XOR	Logical Operations
ADC, ADD, SUB, SBC	Arithmetic Operations

12.1.3 Direct

In Direct instructions, the operands are referenced by their memory address.

The direct addressing mode consists of two sub-modes:

Direct (short)

The address is a byte, thus requires only one byte after the opcode, but only allows 00 - FF addressing space.

Direct (long)

The address is a word, thus allowing 64 Kbyte addressing space, but requires 2 bytes after the opcode.

12.1.4 Indexed (No Offset, Short, Long)

In this mode, the operand is referenced by its memory address, which is defined by the unsigned addition of an index register (X or Y) with an offset.

The indirect addressing mode consists of three sub-modes:

Indexed (No Offset)

There is no offset, (no extra byte after the opcode), and allows 00 - FF addressing space.

Table 51. Instructions supporting Direct, Indexed, Indirect and Indirect Indexed addressing modes (continued)

Long and Short instructions	Function
BCP	Bit Compare
Short Instructions only	**Function**
CLR	Clear
INC, DEC	Increment/Decrement
TNZ	Test Negative or Zero
CPL, NEG	1 or 2 Complement
BSET, BRES	Bit Operations
BTJT, BTJF	Bit Test and Jump Operations
SLL, SRL, SRA, RLC, RRC	Shift and Rotate Operations
SWAP	Swap Nibbles
CALL, JP	Call or Jump subroutine

12.1.7 Relative mode (Direct, Indirect)

This addressing mode is used to modify the PC register value by adding an 8-bit signed offset to it.

Table 52. Instructions supporting relative addressing mode

Available relative direct/Indirect instructions	Function
JRxx	Conditional Jump
CALLR	Call Relative

The relative addressing mode consists of two sub-modes:

Relative (Direct)

The offset follows the opcode.

Relative (Indirect)

The offset is defined in memory, of which the address follows the opcode.

Figure 64. V_{OL} high sink V_{DD}=5 V

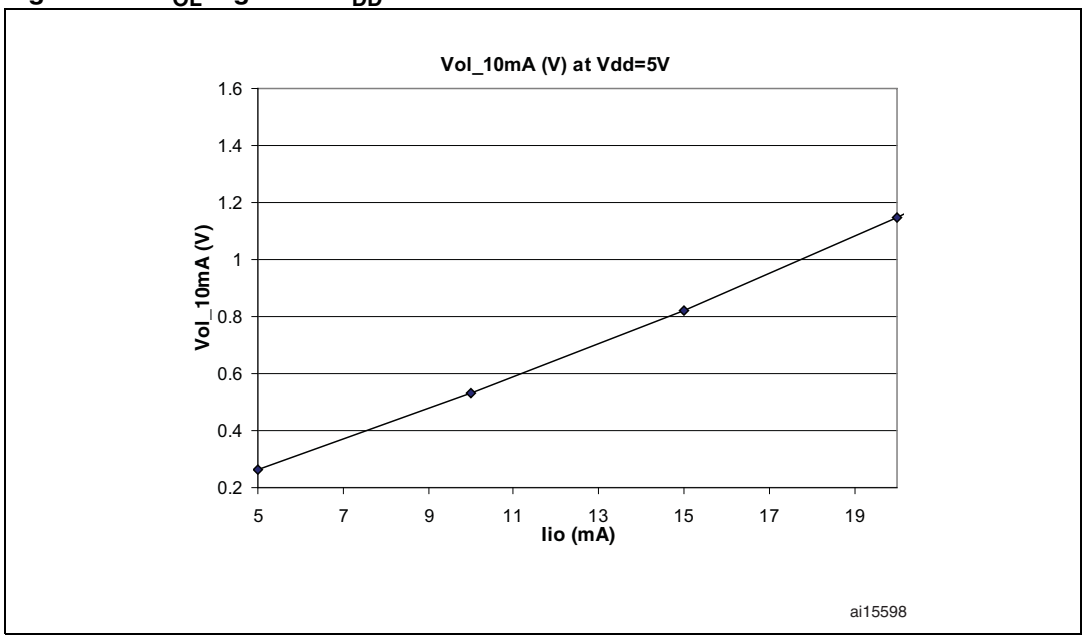


Figure 65. V_{OL} very high sink V_{DD}=5 V

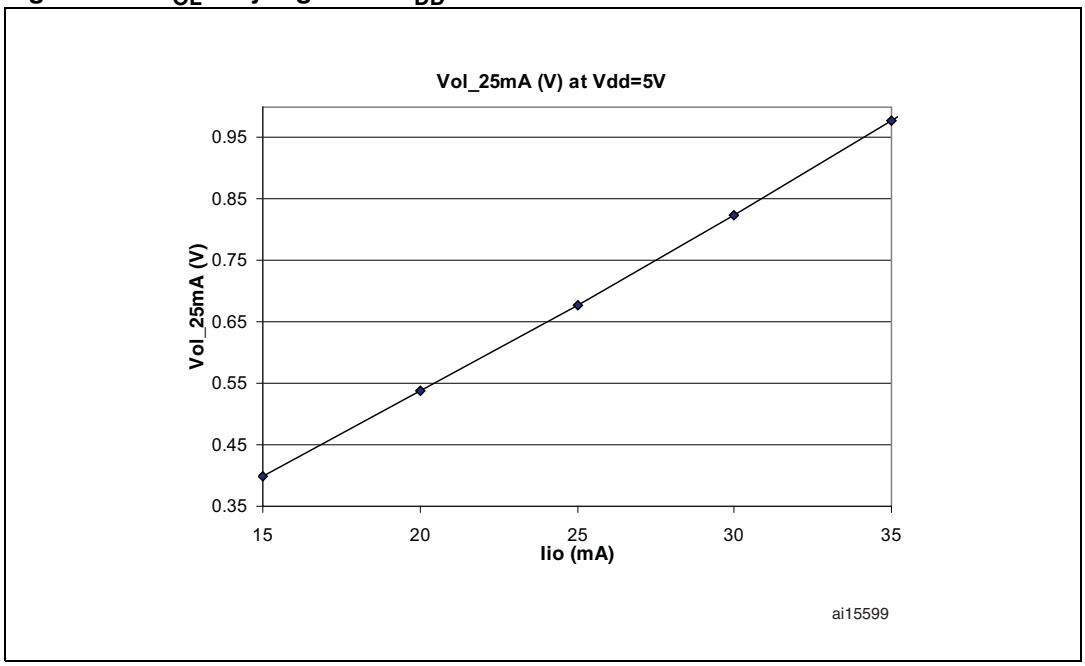


Figure 66. V_{OL} standard vs. V_{DD}

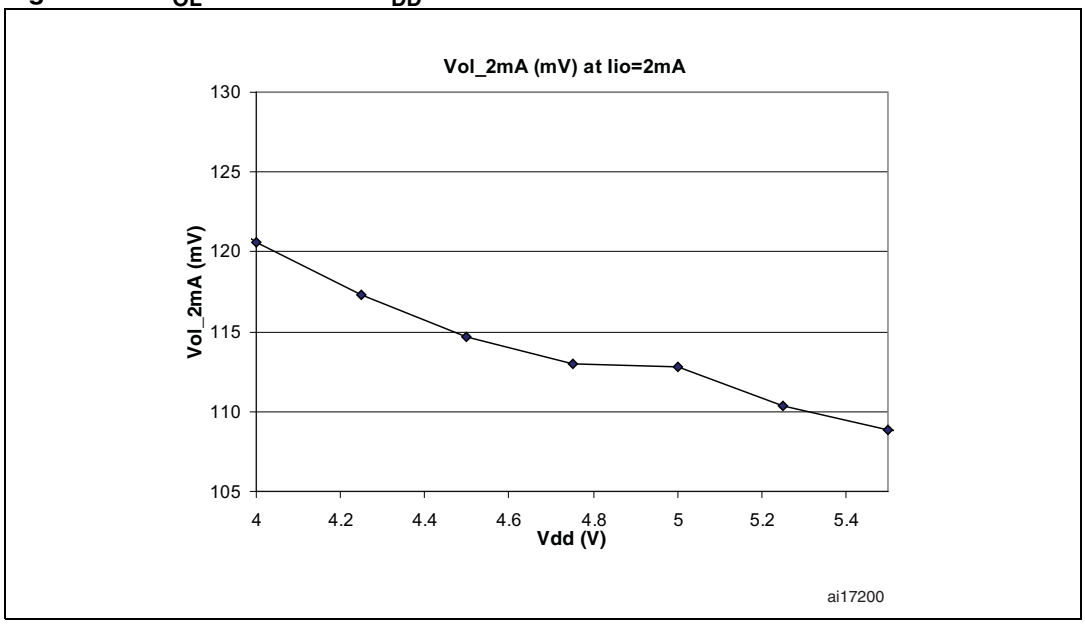


Figure 67. V_{OL} high sink vs. V_{DD}

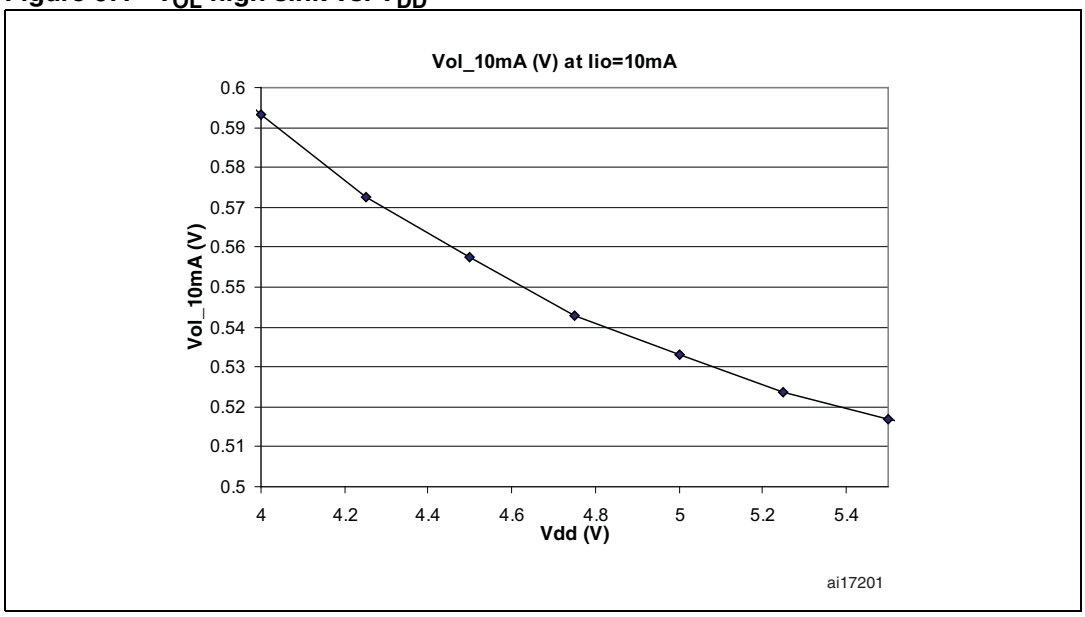


Figure 73 and *Figure 74* show the reset circuit which protects the device against parasitic resets:

- The output of the external reset circuit must have an open-drain output to drive the ST7 reset pad. Otherwise the device can be damaged when the ST7 generates an internal reset (LVD or watchdog).
- Whatever the reset source is (internal or external), the user must ensure that the level on the $\overline{\text{RESET}}$ pin can go below the V_{IL} max. level specified in *Section Table 72.: Asynchronous RESET pin*. Otherwise the reset will not be taken into account internally.
- Because the reset circuit is designed to allow the internal reset to be output in the $\overline{\text{RESET}}$ pin, the user must ensure that the current sunk on the $\overline{\text{RESET}}$ pin is less than the absolute maximum value specified for $I_{INJ(RESET)}$ in *Section Table 56.: Current characteristics*.

When the LVD is enabled:

- It is recommended not to connect a pull-up resistor or capacitor. A 10 nF pull-down capacitor is required to filter noise on the reset line.
- In case a capacitive power supply is used, it is recommended to connect a 1 MΩ pull-down resistor to the $\overline{\text{RESET}}$ pin to discharge any residual voltage induced by the capacitive effect of the power supply (this will add 5 µA to the power consumption of the MCU).
- Tips when using the LVD:
  a) Check that all recommendations related to ICCCLK and reset circuit have been applied (see notes above).
  b) Check that the power supply is properly decoupled (100 nF + 10 µF close to the MCU). Refer to AN1709 and AN2017. If this cannot be done, it is recommended to put a 100 nF + 1 MΩ pull-down on the $\overline{\text{RESET}}$ pin.
  c) The capacitors connected on the $\overline{\text{RESET}}$ pin and also the power supply are key to avoid any start-up marginality. In most cases, steps a) and b) above are sufficient for a robust solution. Otherwise: replace 10 nF pull-down on the $\overline{\text{RESET}}$ pin with a 5 µF to 20 µF capacitor.

Figure 73. $\overline{\text{RESET}}$ pin protection when LVD is enabled

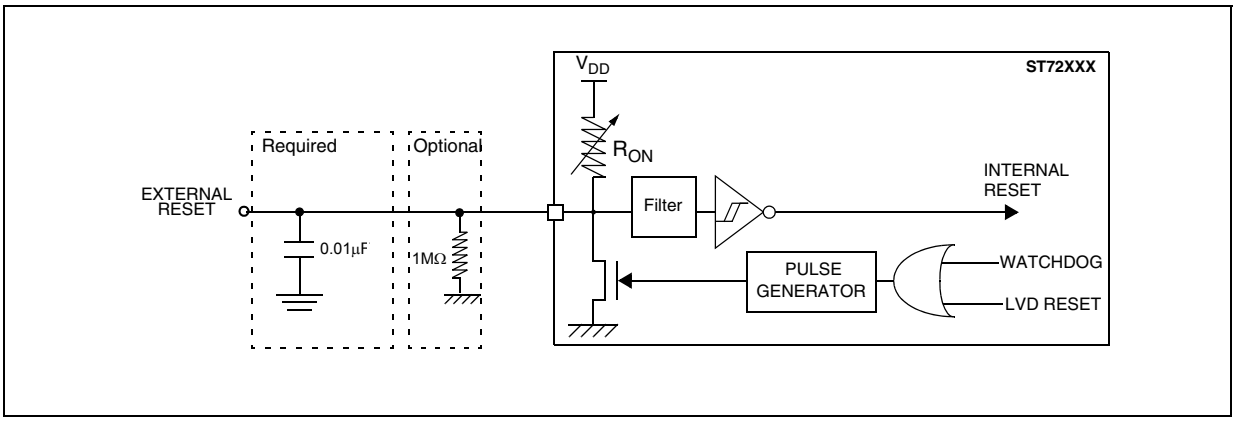


Figure 83. 40-lead very thin fine pitch quad flat no-lead package outline

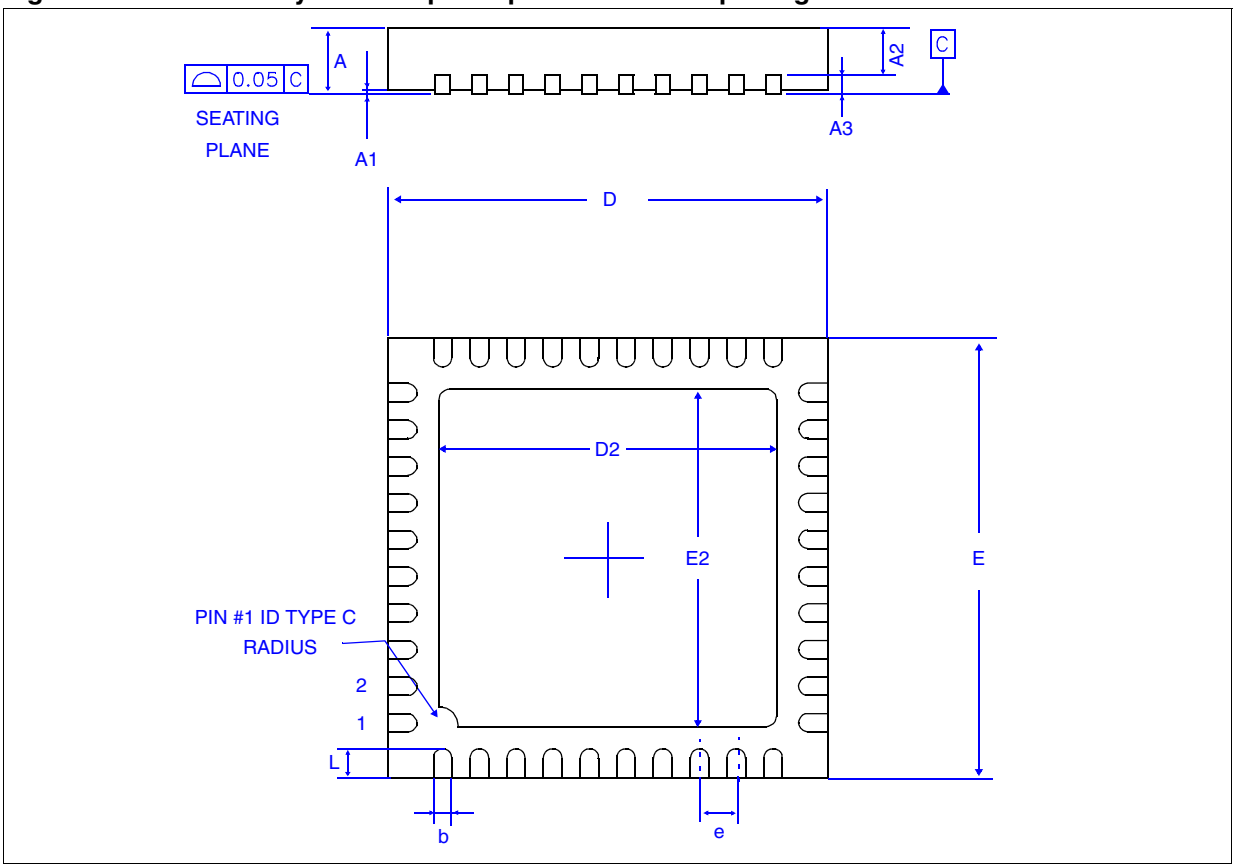


Table 84. 40-lead very thin fine pitch quad flat no-lead package mechanical data

Dim.	mm			inches[1]		
	Min	Typ	Max	Min	Typ	Max
A	0.800	0.900	1.000	0.0315	0.0354	0.0394
A1		0.020	0.050		0.0008	0.0020
A2		0.650	1.000		0.0260	0.0390
A3		0.200			0.0080	
b	0.180	0.250	0.300	0.0070	0.0100	0.0120
D	5.850	6.000	6.150	0.2300	0.2360	0.2420
D2	2.750	2.90	3.050	0.1080	0.1140	0.1200
E	5.850	6.000	6.150	0.2300	0.2360	0.2420
E2	2.750	2.900	3.050	0.1080	0.1140	0.1200
e		0.500			0.0200	
L	0.300	0.400	0.500	0.0120	0.0160	0.0200
	Number of pins					
N	40					

1. Values in inches are converted from mm and rounded to 4 decimal digits.

15.4 ST7 application notes

Table 88. ST7 application notes

Identification	Description
Application examples	
AN1658	Serial Numbering Implementation
AN1720	Managing the Readout 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 971	I²C Communication Between ST7 and M24Cxx EEPROM
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 wakeup 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