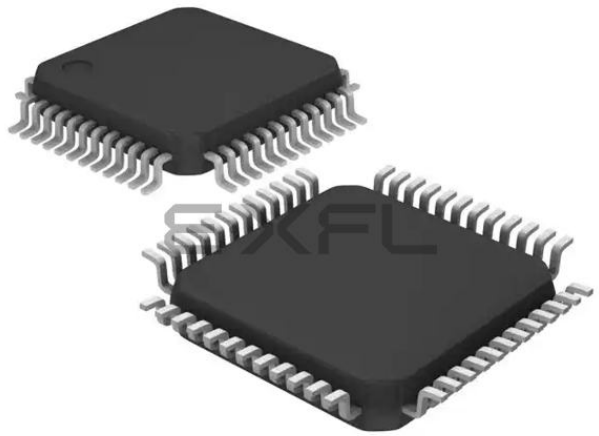


Welcome to E-XFL.COM

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, LINbus, SCI, SPI
Peripherals	LVD, POR, PWM, WDT
Number of I/O	34
Program Memory Size	16KB (16K x 8)
Program Memory Type	FLASH
EEPROM Size	256 x 8
RAM Size	1K x 8
Voltage - Supply (Vcc/Vdd)	2.7V ~ 5.5V
Data Converters	A/D 12x10b
Oscillator Type	Internal
Operating Temperature	-40°C ~ 85°C (TA)
Mounting Type	Surface Mount
Package / Case	48-LQFP
Supplier Device Package	-
Purchase URL	https://www.e-xfl.com/product-detail/stmicroelectronics/st72f345c4t6

1 Description

The ST7234x devices are members of the ST7 microcontroller family. *Table 2* gives the available part numbers and details on the devices. All devices are based on a common industry-standard 8-bit core, featuring an enhanced instruction set.

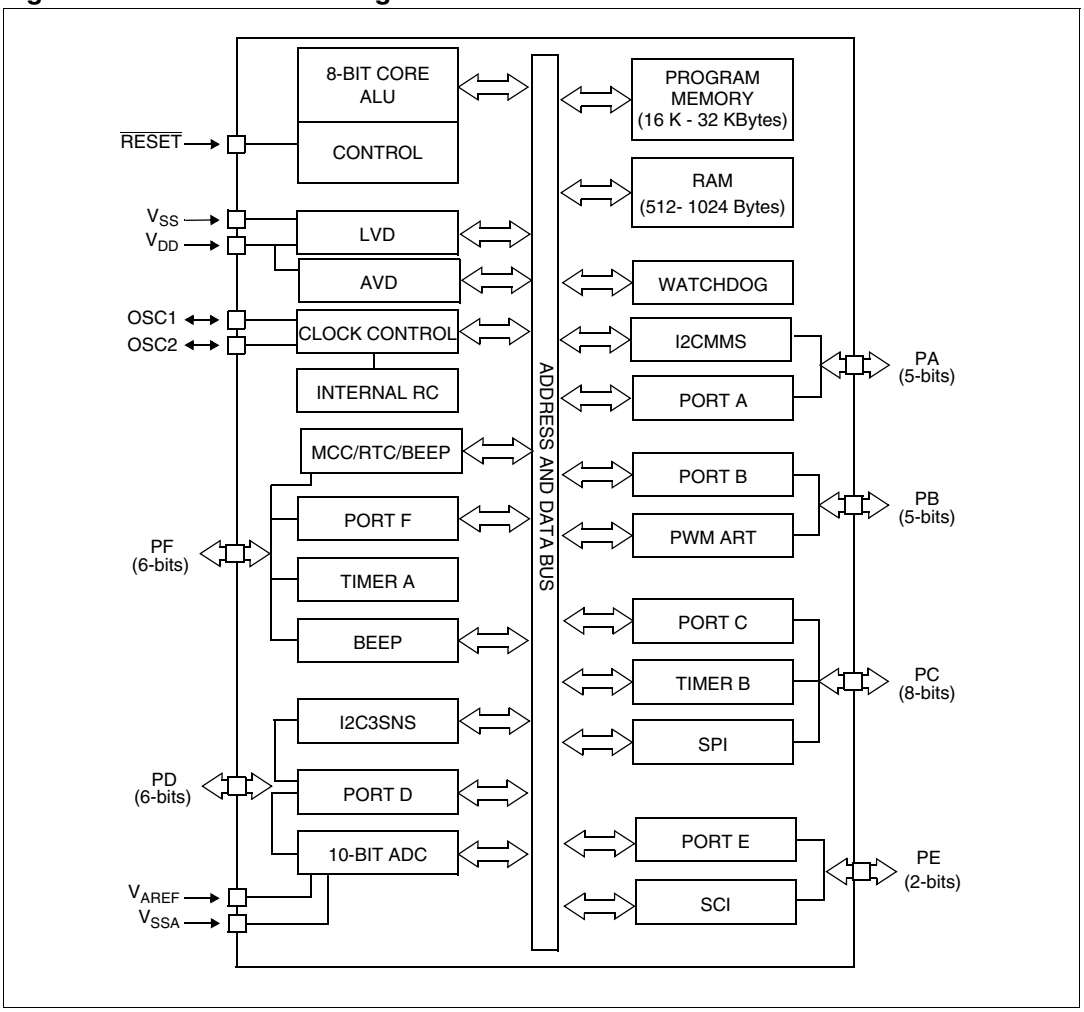
They feature single-voltage Flash memory with byte-by-byte in-circuit programming (ICP) and in-application programming (IAP) capabilities.

Under software control, all devices can be placed in Wait, Slow, Auto-wakeup from Halt, Active-halt or Halt mode, reducing the power consumption when the application is in idle or stand-by state.

The enhanced instruction set and addressing modes of the ST7 offer both power and flexibility to software developers, enabling the design of highly efficient and compact application code. In addition to standard 8-bit data management, all ST7 microcontrollers feature true bit manipulation, 8x8 unsigned multiplication and indirect addressing modes.

The devices feature an on-chip debug module (DM) to support in-circuit debugging (ICD). For a description of the DM registers, refer to the ST7 ICC Protocol Reference Manual.

Figure 1. General block diagram



Legend / Abbreviations for *Table 3*:

Type: I = input, O = output, S = supply

Input level: A = Dedicated analog input

In/Output level: C_T= CMOS $0.3V_{DD}/0.7V_{DD}$ with input trigger

Output level: HS = 20 mA high sink (on N-buffer only)

Port and control configuration:

- Input: float = floating, wpu = weak pull-up, int = interrupt [1], ana = analog
- Output: OD = open drain [2], PP = push-pull

The reset configuration of each pin is shown in bold. This configuration is valid as long as the device is in reset state.

On the chip, each I/O port may have up to 8 pads. Pads that are not bonded to external pins are set in input pull-up configuration after reset through the option byte Package selection. The configuration of these pads must be kept at reset state to avoid added current consumption.

Table 3. Device pin description

Pin n°			Pin name	Type	Level		Port						Main function (after reset)	Alternate function	
							Input [1]				Output				
LQFP32	LQFP44	LQFP48			Input	Output	float	wpu	int	ana	OD	PP			
1	13	14	V_{DDA} [2]	S									Analog supply voltage		
2	14	15	V_{SSA} [2]	S									Analog ground voltage		
3	15	16	PF0/MCO/AIN8	I/O	C_T		X	ei1		X	X	X	Port F0	Main clock out (f_{OSC}/2)	ADC analog input 8
4	16	17	PF1 (HS)/BEEP	I/O	C_T	HS	X	ei1			X	X	Port F1	Beep signal output	
-	17	18	PF2 (HS) [3]	I/O	C_T	HS	X		ei1		X	X	Port F2		
5	18	19	PF4/OCMP1_A/AIN10	I/O	C_T		X	X		X	X	X	Port F4	Timer A output compare 1	ADC analog input 10
6	19	20	PF6 (HS)/ICAP1_A	I/O	C_T	HS	X	X			X	X	Port F6	Timer A Input Capture 1	
7	20	21	PF7 (HS)/EXTCLK_A	I/O	C_T	HS	X	X			X	X	Port F7	Timer A external clock source	
-	21	22	V_{DD_0} [2]	S									Digital main supply voltage		
-	22	23	V_{SS_0} [2]	S									Digital ground voltage		
8	23	24	PC0/OCMP2_B/AIN12	I/O	C_T		X	X		X	X	X	Port C0	Timer B output compare 2	ADC analog input 12
9	24	27	PC1/OCMP1_B/AIN13	I/O	C_T		X	X		X	X	X	Port C1	Timer B output compare 1	ADC analog input 13

5.7 Register description

5.7.1 EEPROM control/status register (EECSR)

Reset value: 0000 0000 (00h)

7							0
0	0	0	0	0	0	E2LAT	E2PGM
Read/Write							

Bits 7:2 = Reserved, forced by hardware to 0.

Bit 1 = **E2LAT** *Latch Access Transfer*

This bit is set by software. It is cleared by hardware at the end of the programming cycle. It can only be cleared by software if the E2PGM bit is cleared.
0: Read mode
1: Write mode

Bit 0 = **E2PGM** *Programming control and status*

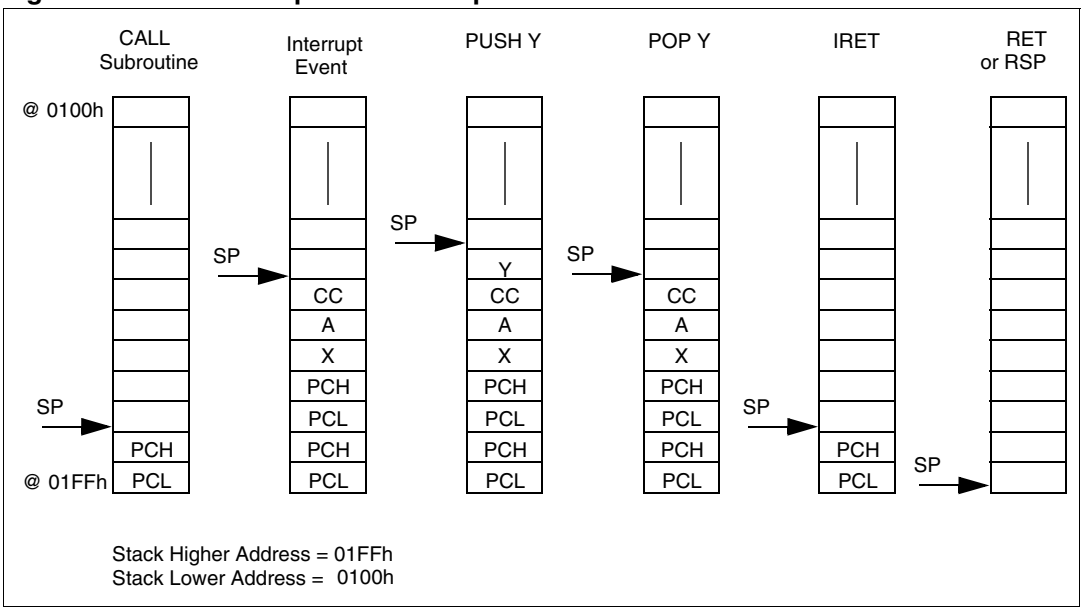
This bit is set by software to begin the programming cycle. At the end of the programming cycle, this bit is cleared by hardware.
0: Programming finished or not yet started
1: Programming cycle is in progress

Note: If the E2PGM bit is cleared during the programming cycle, the memory data is not guaranteed.

Table 5. Data EEPROM register map and reset values

Address (Hex.)	Register label	7	6	5	4	3	2	1	0
0020h	**EECSR** Reset value	0	0	0	0	0	0	E2LAT 0	E2PGM 0

Figure 12. Stack manipulation example



Application notes

The LVDRF flag is not cleared when another reset type occurs (external or watchdog), the LVDRF flag remains set to keep trace of the original failure.
In this case, a watchdog reset can be detected by software while an external reset can not.

Caution: When the LVD is not activated with the associated option byte, the WDGRF flag can not be used in the application.

8.5.2 Interrupt software priority registers (ISPRX)

Reset value: 1111 1111 (FFh)

	7							0
ISPR0	I1_3	I0_3	I1_2	I0_2	I1_1	I0_1	I1_0	I0_0
ISPR1	I1_7	I0_7	I1_6	I0_6	I1_5	I0_5	I1_4	I0_4
ISPR2	I1_11	I0_11	I1_10	I0_10	I1_9	I0_9	I1_8	I0_8
ISPR3	1	1	1	1	I1_13	I0_13	I1_12	I0_12
Read/Write (bits 7:4 of **ISPR3** are read only)								

These four registers contain the interrupt software priority of each interrupt vector.

- Each interrupt vector (except reset and TRAP) has corresponding bits in these registers where its own software priority is stored. This correspondence is shown in the following table.

Table 15. Interrupt vector addresses

Vector address	ISPRx bits
FFFBh-FFFAh	I1_0 and I0_0 bits*
FFF9h-FFF8h	I1_1 and I0_1 bits
...	...
FFE1h-FFE0h	I1_13 and I0_13 bits

- Each I1_x and I0_x bit value in the ISPRx registers has the same meaning as the I1 and I0 bits in the CC register.
- Level 0 can not be written (I1_x=1, I0_x=0). In this case, the previously stored value is kept. (example: previous = CFh, write = 64h, result = 44h)

The reset, and TRAP vectors have no software priorities. When one is serviced, the I1 and I0 bits of the CC register are both set.

Caution: If the I1_x and I0_x bits are modified while the interrupt x is executed, the following behavior has to be considered: If the interrupt x is still pending (new interrupt or flag not cleared) and the new software priority is higher than the previous one, the interrupt x is re-entered. Otherwise, the software priority stays unchanged up to the next interrupt request (after the IRET of the interrupt x).

Table 16. Dedicated interrupt instruction set

Instruction	New description	Function/example	I1	H	I0	N	Z	C
HALT	Entering Halt mode		1		0			
IRET	Interrupt routine return	Pop CC, A, X, PC	I1	H	I0	N	Z	C
JRM	Jump if I1:0=11 (level 3)	I1:0=11 ?						
JRNM	Jump if I1:0<>11	I1:0<>11 ?						
POP CC	Pop CC from the stack	Mem => CC	I1	H	I0	N	Z	C

Output compare 1 high register (OC1HR)

Read/Write
Reset value: 1000 0000 (80h)

This is an 8-bit register that contains the high part of the value to be compared to the CHR register.

7							0
MSB							LSB
Read/Write							

Output compare 1 low register (OC1LR)

Reset value: 0000 0000 (00h)

This is an 8-bit register that contains the low part of the value to be compared to the CLR register.

7							0
MSB							LSB
Read/Write							

Output compare 2 high register (OC2HR)

Reset value: 1000 0000 (80h)

This is an 8-bit register that contains the high part of the value to be compared to the CHR register.

7							0
MSB							LSB
Read/Write							

Output compare 2 low register (OC2LR)

Reset value: 0000 0000 (00h)

This is an 8-bit register that contains the low part of the value to be compared to the CLR register.

7							0
MSB							LSB
Read/Write							

Counter high register (CHR)

Reset value: 1111 1111 (FFh)

This is an 8-bit register that contains the high part of the counter value.

7							0
MSB							LSB
Read-only							

Master mode transmit sequence

When software writes to the SPIDR register, the data byte is loaded into the 8-bit shift register and then shifted out serially to the MOSI pin most significant bit first.

When data transfer is complete:

- The SPIF bit is set by hardware.
- An interrupt request is generated if the SPIE bit is set and the interrupt mask in the CCR register is cleared.

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

1. An access to the SPICSR register while the SPIF bit is set
2. A read to the SPIDR register

Note: While the SPIF bit is set, all writes to the SPIDR register are inhibited until the SPICSR register is read.

Slave mode operation

In slave mode, the serial clock is received on the SCK pin from the master device.

To operate the SPI in slave mode:

1. Write to the SPICSR register to perform the following actions:
   - Select the clock polarity and clock phase by configuring the CPOL and CPHA bits (see *Figure 60*).
     Note that the slave must have the same CPOL and CPHA settings as the master.
   - Manage the $\overline{SS}$ pin as described in *Slave select management* and *Figure 58*.
     If CPHA = 1, $\overline{SS}$ must be held low continuously.
     If CPHA = 0, $\overline{SS}$ must be held low during byte transmission and pulled up between each byte to let the slave write in the shift register.
2. Write to the SPICR register to clear the MSTR bit and set the SPE bit to enable the SPI I/O functions.

Slave mode transmit sequence

When the software writes to the SPIDR register, the data byte is loaded into the 8-bit shift register and then shifted out serially to the MISO pin most significant bit first.

The transmit sequence begins when the slave device receives the clock signal and the most significant bit of the data on its MOSI pin.

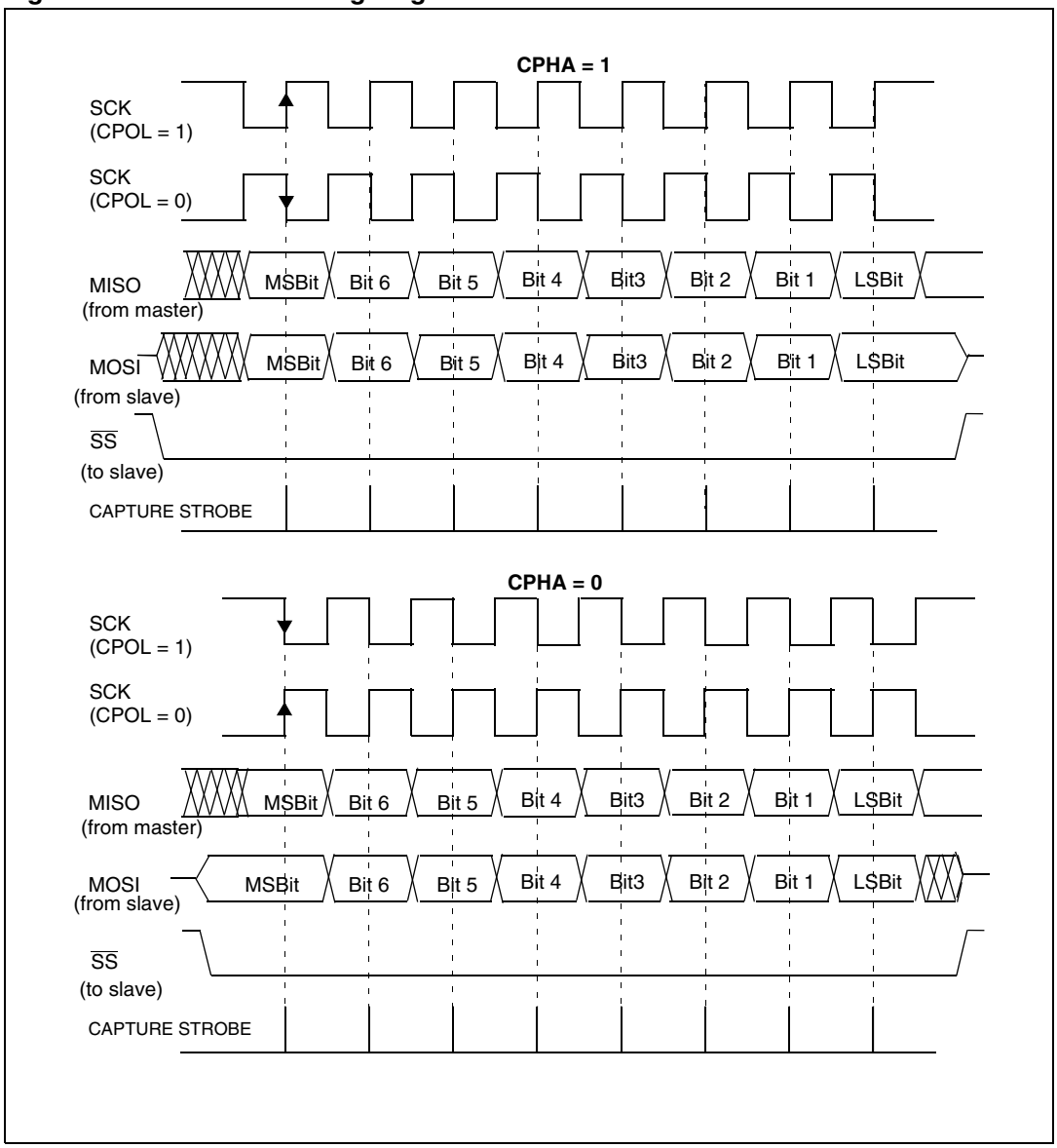
When data transfer is complete:

- The SPIF bit is set by hardware.
- An interrupt request is generated if SPIE bit is set and interrupt mask in the CCR register is cleared.

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

1. An access to the SPICSR register while the SPIF bit is set
2. A write or a read to the SPIDR register

Figure 60. Data clock timing diagram



1. This figure should not be used as a replacement for parametric information. Refer to *Section 13: Electrical characteristics*.

11.4.5 Error flags

Master mode fault (MODF)

Master mode fault occurs when the master device's $\overline{SS}$ pin is pulled low.

When a Master mode fault occurs:

- The MODF bit is set and an SPI interrupt request is generated if the SPIE bit is set.
- The SPE bit is reset. This blocks all output from the device and disables the SPI peripheral.
- The MSTR bit is reset, thus forcing the device into slave mode.

Clearing the MODF bit is done through a software sequence:

1. A read access to the SPICSR register while the MODF bit is set.
2. A write ac cess to the SPICR register.

Note: To avoid any conflicts in an application with multiple slaves, the $\overline{SS}$ pin must be pulled high during the MODF bit clearing sequence. The SPE and MSTR bits may be restored to their original state during or after this clearing sequence.

The hardware does not allow the user to set the SPE and MSTR bits while the MODF bit is set, except in the MODF bit clearing sequence.

In a slave device, the MODF bit can not be set, but in a multimaster configuration the device can be in slave mode with the MODF bit set.

The MODF bit indicates that there might have been a multimaster conflict and allows software to handle this using an interrupt routine and either perform a reset or return to an application default state.

Overrun condition (OVR)

An overrun condition occurs when the master device has sent a data byte and the slave device has not cleared the SPIF bit issued from the previously transmitted byte.

When an Overrun occurs:

- The OVR bit is set, and an interrupt request is generated if the SPIE bit is set.

In this case, the receiver buffer contains the byte sent after the SPIF bit was last cleared. A read to the SPIDR register returns this byte. All other bytes are lost.

The OVR bit is cleared by reading the SPICSR register.

Write collision error (WCOL)

A write collision occurs when the software tries to write to the SPIDR register while a data transfer is taking place with an external device. When this happens, the transfer continues uninterrupted and the software write will be unsuccessful.

Write collisions can occur both in master and slave mode. See also *Slave select management*.

Note: A "read collision" will never occur since the received data byte is placed in a buffer in which access is always synchronous with the CPU operation.

The WCOL bit in the SPICSR register is set if a write collision occurs.

No SPI interrupt is generated when the WCOL bit is set (the WCOL bit is a status flag only).

Clearing the WCOL bit is done through a software sequence (see *Figure 61*).

Setting the RWU bit by software puts the SCI in sleep mode:

- None of the reception status bits can be set.
- All the receive interrupts are inhibited.

A muted receiver can be woken up in one of the following two ways:

- by Idle Line detection if the WAKE bit is reset,
- by Address Mark detection if the WAKE bit is set.

A receiver wakes-up by Idle Line detection when the Receive line has recognized an Idle Frame. Then the RWU bit is reset by hardware but the IDLE bit is not set.

A receiver wakes up by Address Mark detection when it received a “1” as the most significant bit of a word, thus indicating that the message is an address. The reception of this particular word wakes up the receiver, resets the RWU bit and sets the RDRF bit, which allows the receiver to receive this word normally and to use it as an address word.

Parity control

Parity control (generation of parity bit in transmission and parity checking in reception) can be enabled by setting the PCE bit in the SCICR1 register. Depending on the frame length defined by the M bit, the possible SCI frame formats are as listed in *Table 56*.

Table 56. Frame formats

M bit	PCE bit	SCI frame(1)
0	0	| SB | 8 bit data | STB |
	1	| SB | 7-bit data | PB | STB |
1	0	| SB | 9-bit data | STB |
	1	| SB | 8-bit data PB | STB |

1. SB: Start Bit, STB: Stop Bit, PB: Parity Bit.

Note: In case of wakeup by an address mark, the MSB bit of the data is taken into account and not the parity bit

Even parity: The parity bit is calculated to obtain an even number of “1s” inside the frame made of the 7 or 8 LSB bits (depending on whether M is equal to 0 or 1) and the parity bit.

Example: data = 00110101; 4 bits set => parity bit is 0 if even parity is selected (PS bit = 0).

Odd parity: The parity bit is calculated to obtain an odd number of “1s” inside the frame made of the 7 or 8 LSB bits (depending on whether M is equal to 0 or 1) and the parity bit.

Example: data = 00110101; 4 bits set => parity bit is 1 if odd parity is selected (PS bit = 1).

Transmission mode: If the PCE bit is set, then the MSB bit of the data written in the data register is not transmitted but is changed by the parity bit.

Reception mode: If the PCE bit is set, then the interface checks if the received data byte has an even number of “1s” if even parity is selected (PS = 0) or an odd number of “1s” if odd parity is selected (PS = 1). If the parity check fails, the PE flag is set in the SCISR register and an interrupt is generated if PIE is set in the SCICR1 register.

Note: In 10-bit addressing mode, to switch the master to Receiver mode, the software must generate a repeated Start condition and resend the header sequence with the least significant bit set (11110xx1).

Master receiver

Following the address transmission and after SR1 and CR registers have been accessed, the master 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 by hardware 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 69* Transfer sequencing EV7).

To close the communication: before reading the last byte from the DR register, set the STOP bit to generate the Stop condition. The interface goes automatically back to slave mode (M/SL bit cleared).

Note: In order to generate the non-acknowledge pulse after the last received data byte, the ACK bit must be cleared just before reading the second last data byte.

Master transmitter

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

The master waits for a read of the SR1 register followed by a write in the DR register, **holding the SCL line low** (see *Figure 69* Transfer sequencing EV8).

When the acknowledge bit is received, the interface sets:

- EVF and BTF bits with an interrupt if the ITE bit is set.

To close the communication: after writing the last byte to the DR register, set the STOP bit to generate the Stop condition. The interface goes automatically back to slave mode (M/SL bit cleared).

Error cases

- **BERR**: Detection of a Stop or a Start condition during a byte transfer. In this case, the EVF and BERR bits are set by hardware with an interrupt if ITE is set.
  Note that BERR will not be set if an error is detected during the first pulse of each 9-bit transaction:
  Single Master Mode
  If a Start or Stop is issued during the first pulse of a 9-bit transaction, the BERR flag will not be set and transfer will continue however the BUSY flag will be reset. To work around this, slave devices should issue a NACK when they receive a misplaced Start or Stop. The reception of a NACK or BUSY by the master in the middle of communication gives the possibility to re-initiate transmission.
  Multimaster Mode
  Normally the BERR bit would be set whenever unauthorized transmission takes place while transfer is already in progress. However, an issue will arise if an external master generates an unauthorized Start or Stop while the I^2C master is on the first pulse of a 9-bit transaction. It is possible to work around this by polling the BUSY bit during I^2C

Figure 69). BTF is cleared by reading SR1 register followed by writing the next byte in DR register.

- Following a byte reception, this bit is set after transmission of the acknowledge clock pulse if ACK=1. BTF is cleared by reading SR1 register followed by reading the byte from DR register.

  The SCL line is held low while BTF=1.

  0: Byte transfer not done
  1: Byte transfer succeeded

Bit 2 = **ADSL** *Address matched (Slave mode).*

This bit is set by hardware as soon as the received slave address matched with the OAR register content or a general call is recognized. An interrupt is generated if ITE=1. It is cleared by software reading SR1 register or by hardware when the interface is disabled (PE=0).

The SCL line is held low while ADSL=1.

0: Address mismatched or not received
1: Received address matched

Bit 1 = **M/SL** *Master/Slave.*

This bit is set by hardware as soon as the interface is in Master mode (writing START=1). It is cleared by hardware after detecting a Stop condition on the bus or a loss of arbitration (ARLO=1). It is also cleared when the interface is disabled (PE=0).

0: Slave mode
1: Master mode

Bit 0 = **SB** *Start bit (Master mode).*

This bit is set by hardware as soon as the Start condition is generated (following a write START=1). An interrupt is generated if ITE=1. It is cleared by software reading SR1 register followed by writing the address byte in DR register. It is also cleared by hardware when the interface is disabled (PE=0).

0: No Start condition
1: Start condition generated

I²C status register 2 (SR2)

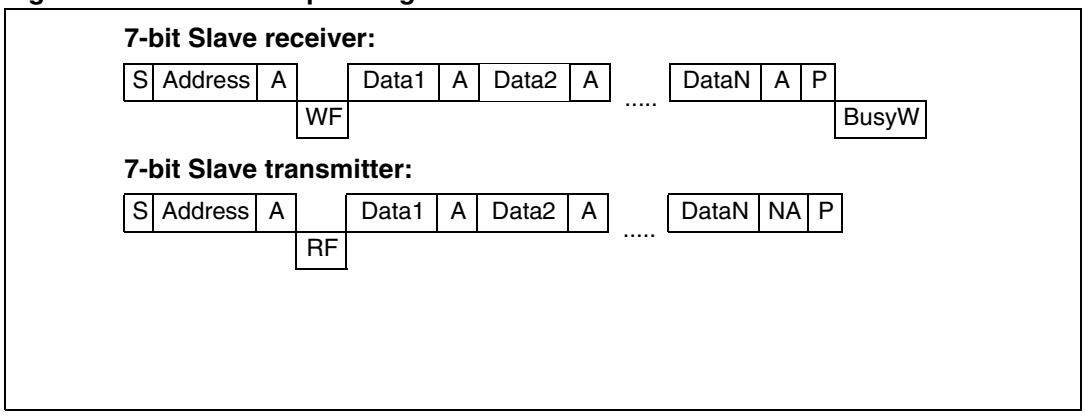
Reset value: 0000 0000 (00h)

7							0
0	0	0	AF	STOPF	ARLO	BERR	GCAL
Read Only							

Bit 7:5 = Reserved.

Forced to 0 by hardware.

Figure 75. Transfer sequencing



Legend: S = Start, P = Stop, A = Acknowledge, NA = Non-acknowledge,
WF = WF event, WFx bit is set (with interrupt if ITWEx=1, after Stop or Restart conditions), cleared by reading the I2C3SSR register while no communication is ongoing.

RF = RF event, RFx is set (with interrupt if ITREx=1, after Stop or Restart conditions), cleared by reading the I2C3SSR register while no communication is ongoing.

BusyW = BusyW flag in the I2C3CR2 register set, cleared by software writing 0.

Note: The I2C3S supports a repeated start (S_r) in place of a stop condition (P).

Figure 76. Byte write

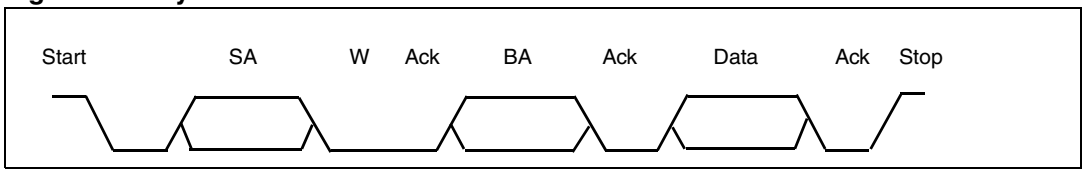


Figure 77. Page write

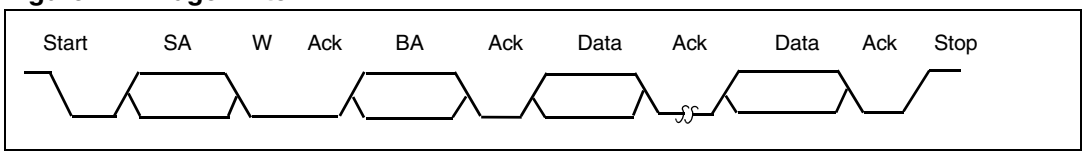


Figure 78. Current address read

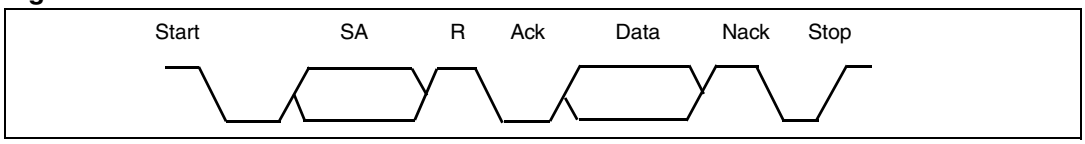
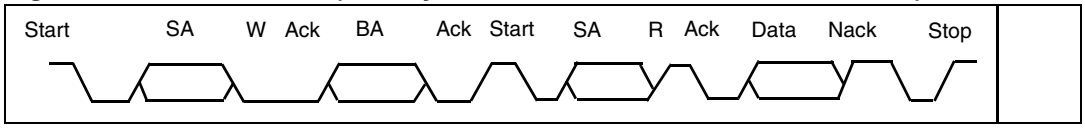


Figure 79. Random read (dummy write + restart + current address read)



I2C slave 1 memory current address register (I2C3SCAR1)

Reset value: 0000 0000 (00h)

7							0
CA7	CA6	CA5	CA4	CA3	CA2	CA1	CA0
Read only							

Bit 7:0 = **CA[7:0]** *Current address of Slave 1 buffer*

This register contains the 8 bit offset of Slave Address 1 reserved area in RAM. It is also cleared by hardware when the interface is disabled (PE =0).

I2C slave 2 memory current address register (I2C3SCAR2)

Reset value: 0000 0000 (00h)

7							0
CA7	CA6	CA5	CA4	CA3	CA2	CA1	CA0
Read only							

Bit 7:0 = **CA[7:0]** *Current address of Slave 2 buffer*

This register contains the 8-bit offset of Slave Address 2 reserved area in RAM. It is also cleared by hardware when the interface is disabled (PE =0).

I2C slave 3 memory current address register (I2C3SCAR3)

Reset value: 0000 0000 (00h)

7							0
CA7	CA6	CA5	CA4	CA3	CA2	CA1	CA0
Read only							

Bit 6:0 = **CA[6:0]** *Current address of Slave 3 buffer*

This register contains the 8-bit offset of slave address 3 reserved area in RAM. It is also cleared by hardware when the interface is disabled (PE =0).

Note: Slave address 3 can store only 128 bytes. For slave address 3, CA7 bit will remain 0. i.e. if the Byte Address sent is 0x80, then the Current Address register will hold the 0x00 value due to an overflow.

Table 71. I^2C3S register map

Address (Hex.)	Register name	7	6	5	4	3	2	1	0
0060h	I2C3SCR1	PL1	PL0	0	ITER	ITRE3	ITRE1/2	ITWE3	ITWE1/2
0061h	I2C3SCR2	0	0	0	WP2	WP1	PE	BusyW	B/W
0062h	I2C3SSR	NACK	BERR	WF3	WF2	WF1	RF3	RF2	RF1
0063h	I2C3SBCR	NB7	NB6	NB5	NB4	NB3	NB2	NB1	NB1
0064h	I2C3SSAR1	ADDR7	ADDR6	ADDR5	ADDR4	ADDR3	ADDR2	ADDR1	EN1
0065h	I2C3SCAR1	CA 7 .. CA0							

Figure 99. Typical V_{OL} at V_{DD} = 2.4 V (std I/Os)

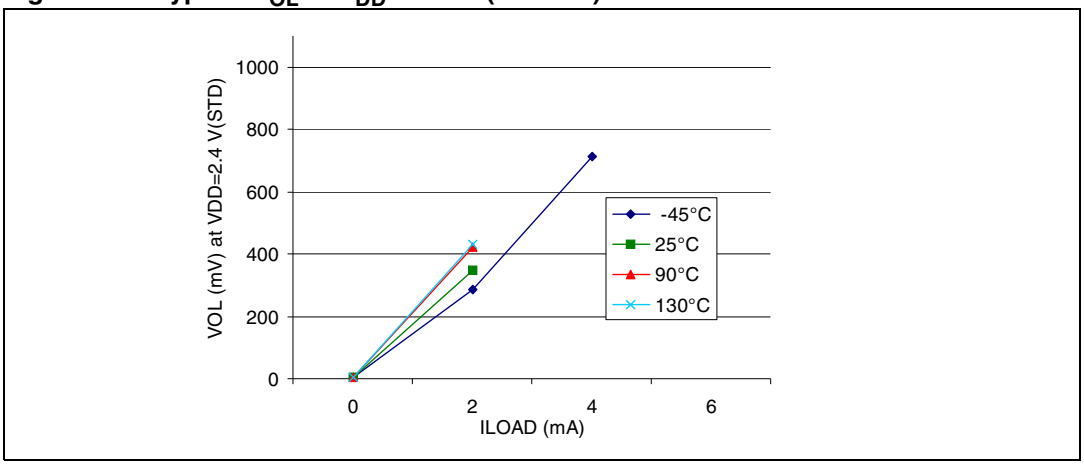


Figure 100. Typical V_{OL} at V_{DD} = 3 V (std I/Os)

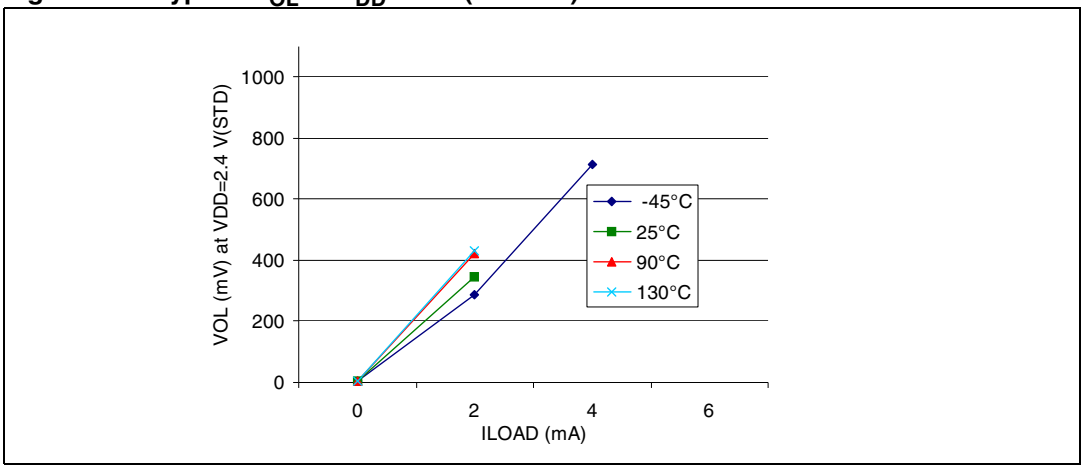


Figure 101. Typical V_{OL} at V_{DD} = 5 V (std I/Os)

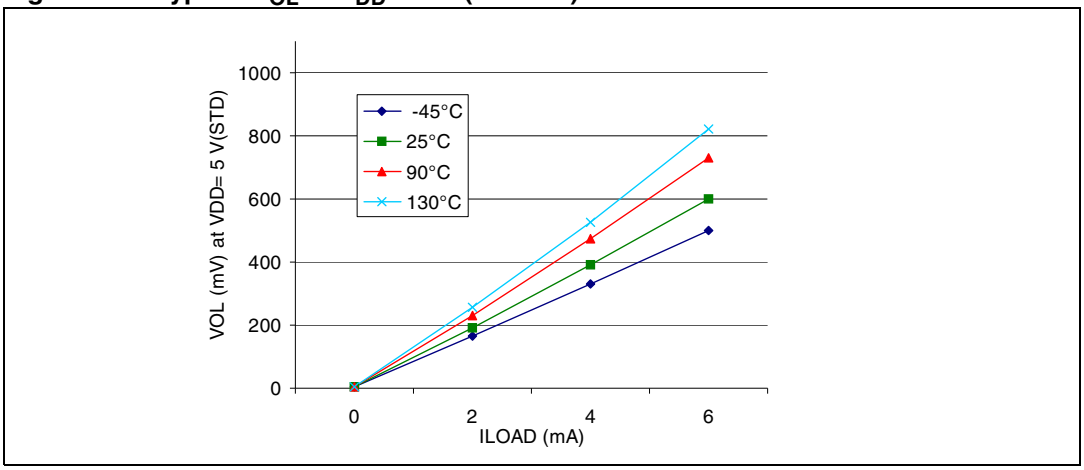
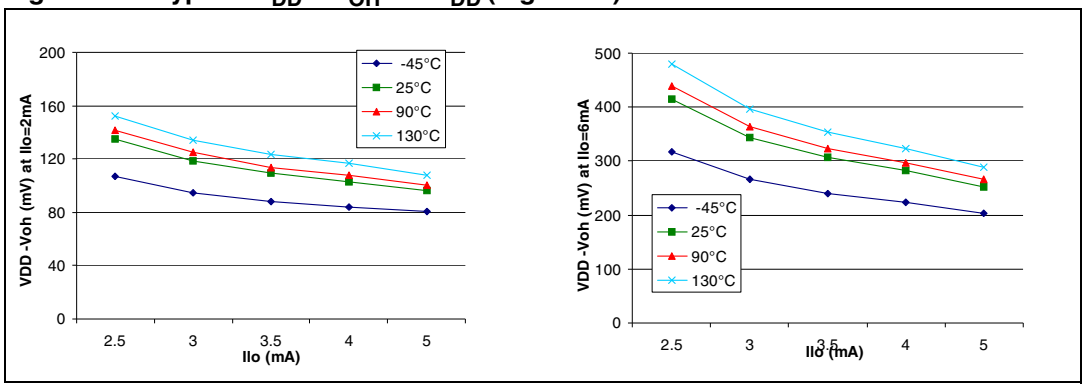


Figure 114. Typical $V_{DD} - V_{OH}$ vs. V_{DD} (high sink)



13.10 Control pin characteristics

13.10.1 Asynchronous $\overline{\text{RESET}}$ pin

T_A = -40 °C to 85 °C, unless otherwise specified.

Table 109. Asynchronous $\overline{\text{RESET}}$ pin characteristics

Symbol	Parameter	Conditions		Min	Typ	Max	Unit
V_{IL}	Input low level voltage			V_{ss} - 0.3		$0.3V_{DD}$	V
V_{IH}	Input high level voltage			$0.7V_{DD}$		V_{DD} + 0.3	
V_{hys}	Schmitt trigger voltage hysteresis (1)				2		V
V_{OL}	Output low level voltage (2)	V_{DD} = 5 V	I_{IO} = +5 mA		0.5	1.0	V
			I_{IO} = +2 mA		0.2	0.4	
R_{ON}	Pull-up equivalent resistor (3)(1)	V_{DD} = 5 V		20	40	80	kΩ
		V_{DD} = 3 V		40	70	120	
$t_{w(RSTL)out}$	Generated reset pulse duration	Internal reset sources		14 (1)	32		µs
$t_{h(RSTL)in}$	External reset pulse hold time (4)			12 (1)			µs
$t_{g(RSTL)in}$	Filtered glitch duration				200		ns

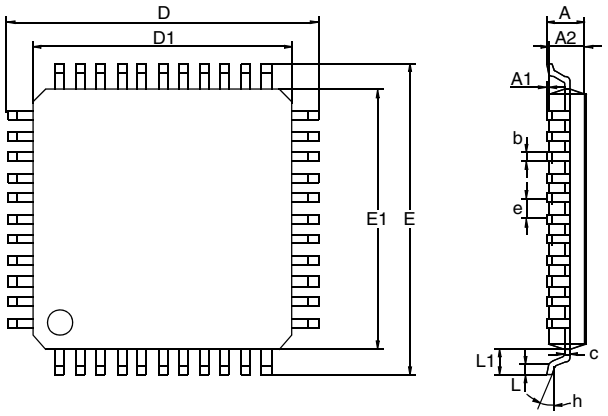
1. Data based on characterization results, not tested in production.
2. The I_{IO} current sunk must always respect the absolute maximum rating specified in *Table 85: Current characteristics* and the sum of I_{IO} (I/O ports and control pins) must not exceed I_{VSS}.
3. The R_{ON} pull-up equivalent resistor is based on a resistive transistor. Specified for voltages on $\overline{\text{RESET}}$ pin between V_{ILmax} and V_{DD}
4. To guarantee the reset of the device, a minimum pulse has to be applied to the $\overline{\text{RESET}}$ pin. All short pulses applied on $\overline{\text{RESET}}$ pin with a duration below $t_{h(RSTL)in}$ can be ignored.

Table 115. 40-lead very thin fine pitch quad flat no-lead package mechanical data

Dim.	mm			inches (1)		
	Min	Typ	Max	Min	Typ	Max
A	0.80	0.90	1.00	0.0315	0.0354	0.0394
A1		0.02	0.05		0.0008	0.0020
A2		0.65	1.00		0.0256	0.0394
A3		0.20			0.0079	
b	0.18	0.25	0.30	0.0071	0.0098	0.0118
D	5.85	6.00	6.15	0.2303	0.2362	0.2421
D2	2.75	2.9	3.05	0.1083	0.1142	0.1201
E	5.85	6	6.15	0.2303	0.2362	0.2421
E2	2.75	2.9	3.05	0.1083	0.1142	0.1201
e		0.50			0.0197	
L	0.30	0.40	0.50	0.0118	0.0157	0.0197
N	Number of pins					
	40					

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

Figure 121. 44-pin low profile quad flat package outline



1. Drawing is not to scale.

```
OUT: RIM
JP while_loop
.call_routine; entry to call_routine
PUSH A
PUSH X
PUSH CC
.ext1_rt; entry to interrupt routine
LD A,#$00
LD sema,A
IRET
```

16.1.2 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.

16.2 Clearing active interrupts outside interrupt routine

When an active interrupt request occurs at the same time as the related flag is being cleared, an unwanted reset may occur.

Note: Clearing the related interrupt mask will not generate an unwanted reset.

Concurrent interrupt context

The symptom does not occur when the interrupts are handled normally, i.e. when:

- The interrupt flag is cleared within its own interrupt routine
- The interrupt flag is cleared within any interrupt routine
- The interrupt flag is cleared in any part of the code while this interrupt is disabled

If these conditions are not met, the symptom can be avoided by implementing the following sequence:

Perform SIM and RIM operation before and after resetting an active interrupt request.

Example:

```
SIM
reset interrupt flag
RIM
```

Nested interrupt context:

The symptom does not occur when the interrupts are handled normally, i.e. when:

- The interrupt flag is cleared within its own interrupt routine
- The interrupt flag is cleared within any interrupt routine with higher or identical priority level
- The interrupt flag is cleared in any part of the code while this interrupt is disabled