



# ST52F510/F513/F514

## 8-BIT INTELLIGENT CONTROLLER UNIT (ICU)

Two Timer/PWMs, ADC, I<sup>2</sup>C, SPI, SCI

### TARGET SPECIFICATION

#### Memories

- Up to 8 Kbytes Single Voltage Flash Memory
- 256 bytes of Register File
- 256 bytes of RAM
- Up to 4 Kbytes Data EEPROM
- In Situ Programming in Flash devices (ISP)
- Single byte and Page modes and In Application Programming for writing data in Flash memory
- Readout protection and flexible write protection

#### Core

- Register File based architecture
- 107 basic instructions
- Hardware multiplication and division
- Decision Processor for the implementation of Fuzzy Logic algorithms
- Deep System and User Stacks

#### Clock and Power Supply

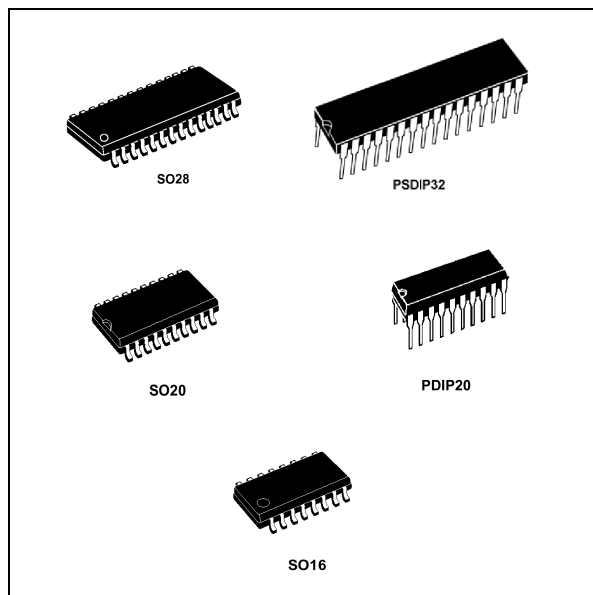
- Up to 24 MHz clock frequency
- Programmable Oscillator modes:
  - 10 MHz Internal Oscillator
  - External Clock/ Oscillator
  - External RC Oscillator
- Power-On Reset (POR)
- Programmable Low Voltage Detector (PLVD) with 3 configurable thresholds
- Power Saving features

#### Interrupts

- 8 interrupt vectors with one SW Trap
- Non-Maskable Interrupt (NMI)
- Two Port Interrupts with up to 16 sources

#### I/O Ports

- From 10 up to 22 I/O PINs configurable in pull-up, push-pull, weak pull-up, open-drain and high-impedance
- High current sink/source in all pins



#### Peripherals

- On-chip 10-bit A/D Converter with 8 channel analog multiplexer and Autocalibration.
- 2 Programmable 16 bit Timer/PWMs with internal 16-bit Prescaler featuring:
  - PWM output
  - Input capture
  - Output compare
  - Pulse generator mode
- Watchdog timer
- Serial Communication Interface (SCI) with asynchronous protocol (UART).
- I<sup>2</sup>C™ Peripheral with master and slave mode
- 3-wire SPI™ Peripheral supporting Single Master and Multi Master SPI modes

#### Development tools

- High level Software tools
- 'C' Compiler
- Emulator
- Low cost Programmer
- Gang Programmer



---

# TABLE OF CONTENTS

---

<b>1 GENERAL DESCRIPTION</b>	<b>7</b>
1.1 Introduction	7
1.2 Functional Description	8
1.2.1 Memory Programming Mode	8
1.2.2 Working Mode	8
1.3 Pin Description	16
<b>2 INTERNAL ARCHITECTURE</b>	<b>17</b>
2.1 Control Unit and Data Processing Unit	17
2.1.1 Program Counter	18
2.1.2 Flags	18
2.2 Arithmetic Logic Unit	19
2.3 Register Description	20
<b>3 ADDRESSING SPACES</b>	<b>21</b>
3.1 Memory Interface	21
3.2 Register File	21
3.3 Program/Data Memory	21
3.4 System and User Stacks	23
3.5 Input Registers	24
3.6 Output registers	24
3.7 Configuration Registers & Option Bytes	25
<b>4 MEMORY PROGRAMMING</b>	<b>31</b>
4.1 Program/Data Memory Organization	31
4.2 Memory Programming	32
4.2.1 Programming Mode start	32
4.2.2 Fast Programming procedure	33
4.2.3 Random data writing	33
4.2.4 Option Bytes Programming	34
4.3 Memory Verify	35
4.3.1 Fast read procedure	35
4.3.2 Random data reading	36
4.4 Memory Lock	36
4.5 ID Code	37
4.6 Error cases	37
4.7 In-Situ Programming (ISP)	38
4.8 In-Application Programming (IAP)	38
4.8.1 Single byte write	38
4.8.2 Block write	38
4.8.3 Memory Corruption Prevention	38
4.8.4 Option Bytes	39
4.8.5 Input Register	39

**5 INTERRUPTS . . . . . 40**

5.1 Interrupt Processing . . . . . 40

5.2 Global Interrupt Request Enabling . . . . . 40

5.3 Interrupt Sources . . . . . 41

5.4 Interrupt Maskability and Priority Levels . . . . . 41

5.5 Interrupt RESET . . . . . 41

5.6 Register Description . . . . . 42

**6 CLOCK, RESET & POWER SAVING MODES . . . . . 44**

6.1 Clock. . . . . 44

6.2 Reset. . . . . 45

6.2.1 External Reset . . . . . 45

6.2.2 Reset Procedures. . . . . 45

6.3 Programmable Low Voltage Detector. . . . . 46

6.4 Power Saving modes . . . . . 46

6.4.1 Wait Mode. . . . . 46

6.4.2 Halt Mode . . . . . 46

6.5 Register Description . . . . . 48

6.5.1 Configuration Register . . . . . 48

6.5.2 Option Bytes . . . . . 48

**7 I/O PORTS . . . . . 50**

7.1 Introduction . . . . . 50

7.2 Input Mode . . . . . 50

7.3 Output Mode . . . . . 50

7.4 Interrupt Mode. . . . . 50

7.5 Alternate Functions. . . . . 51

7.6 Register Description . . . . . 51

7.6.1 Configuration Registers . . . . . 52

7.6.2 Input Registers . . . . . 54

7.6.3 Output Registers. . . . . 55

**8 FUZZY COMPUTATION (DP). . . . . 56**

8.1 Fuzzy Inference . . . . . 56

8.2 Fuzzyfication Phase . . . . . 56

8.3 Inference Phase . . . . . 56

8.4 Defuzzyfication . . . . . 57

8.5 Input Membership Function . . . . . 57

8.6 Output Singleton. . . . . 58

8.7 Fuzzy Rules . . . . . 58

**9 INSTRUCTION SET . . . . . 60**

9.1 Addressing Modes . . . . . 60

9.2 Instruction Types. . . . . 60

**10 10-bit A/D CONVERTER . . . . . 65**

10.1 Introduction . . . . . 65



10.2 Functional Description .....	66
10.3 Operating Modes .....	66
10.3.1 One Channel Single Mode .....	66
10.3.2 Multiple Channels Single Mode .....	66
10.3.3 One Channel Continuous Mode .....	66
10.3.4 Multiple Channels Continuous Mode .....	67
10.4 Power Down Mode .....	67
10.5 A/D Converter Register Description .....	67
10.5.1 A/D Converter Configuration Registers .....	67
10.5.2 Input Registers .....	68
<b>11 WATCHDOG TIMER.....</b>	<b>69</b>
11.1 Functional Description .....	69
11.2 Register Description .....	69
<b>12 PWM/TIMERS.....</b>	<b>71</b>
12.1 Introduction .....	71
12.2 Timer Mode .....	71
12.3 PWM Mode .....	73
12.3.1 Simultaneous Start .....	74
12.4 Timer Interrupts .....	74
12.5 PWM/Timer 0 Register Description .....	74
12.5.1 PWM/Timer 0 Configuration Registers .....	74
12.5.2 PWM/Timer 0 Input Registers .....	75
12.5.3 PWM/Timer 0 Output Registers .....	76
12.6 PWM/Timer 1 Register Description .....	77
12.6.1 PWM/Timer 1 Configuration Registers .....	77
12.6.2 PWM/Timer 1 Input Registers .....	78
12.6.3 PWM/Timer 1 Output Registers .....	79
<b>13 SERIAL COMMUNICATION INTERFACE.....</b>	<b>80</b>
13.1 SCI Receiver block .....	80
13.1.1 Recovery Buffer Block .....	81
13.1.2 SCDR_RX Block .....	81
13.2 SCI Transmitter Block .....	82
13.3 Baud Rate Generator Block .....	82
13.4 SCI Register Description .....	84
13.4.1 SCI Configuration Registers .....	84
13.4.2 SCI Input Registers .....	85
13.4.3 SCI Output Register .....	85
<b>14 I2C BUS INTERFACE (I2C).....</b>	<b>86</b>
14.1 Introduction .....	86
14.2 Main Features .....	86
14.3 General Description .....	86

- 14.3.1 Mode Selection . . . . . 86
- 14.3.2 Communication Flow . . . . . 86
- 14.3.3 SDA/SCL Line Control . . . . . 87
- 14.4 Functional Description . . . . . 87
  - 14.4.1 Slave Mode . . . . . 87
  - 14.4.2 Master Mode . . . . . 88
- 14.5 Register Description . . . . . 92
  - 14.5.1 I2C Interface Configuration Registers . . . . . 92
  - 14.5.2 I2C Interface Input Registers . . . . . 93
  - 14.5.3 I2C Interface Output Registers . . . . . 95
- 15 SERIAL PERIPHERAL INTERFACE (SPI) . . . . . 96**
  - 15.1 Introduction . . . . . 96
  - 15.2 Main Features . . . . . 96
  - 15.3 General description . . . . . 96
  - 15.4 Functional Description . . . . . 96
    - 15.4.1 Master Configuration . . . . . 96
    - 15.4.2 Slave Configuration . . . . . 98
    - 15.4.3 Data Transfer Format . . . . . 98
    - 15.4.4 Write Collision Error . . . . . 98
    - 15.4.5 Master Mode Fault . . . . . 99
    - 15.4.6 Overrun Condition . . . . . 101
    - 15.4.7 Single Master and Multimaster Configurations . . . . . 101
    - 15.4.8 Interrupts . . . . . 102
  - 15.5 SPI Register Description . . . . . 103
    - 15.5.1 SPI Configuration Registers . . . . . 103
    - 15.5.2 SPI Input Register . . . . . 104
    - 15.5.3 SPI Output Register . . . . . 105

## 1 GENERAL DESCRIPTION

### 1.1 Introduction

ST52F510/F513/F514 are devices of ST FIVE family of 8-bit Intelligent Controller Units (ICU), which can perform, both boolean and Fuzzy algorithms in an efficient manner, in order to reach the best performances that the two methodologies allow.

Produced by STMicroelectronics using the reliable high performance CMOS process for Single Voltage Flash versions, ST52F510/F513/F514 include integrated on-chip peripherals that allow maximization of system reliability, and decreased system costs in order to minimize the number of external components.

The flexible I/O configuration of ST52F510/F513/F514 allow one to interface with a wide range of external devices (for example D/A converters or power control devices), and to communicate with the most common serial standards.

ST52F510/F513/F514 pins are configurable. The user can set input or output signals on each single pin in 8 different modes, reducing the need for external components in order to supply a suitable interface with the port pins.

A hardware multiplier and divider, together with a wide instruction set, allow the implementation of complex functions by using a single instruction. Therefore, program memory utilization and computational speed is optimized.

Fuzzy Logic dedicated structures in ST52F510/F513/F514 ICU's can be exploited to model complex system with high accuracy in a useful and simple manner.

Fuzzy Expert Systems for overall system management and Fuzzy Real time Controls can be designed to increase performance at competitive costs.

The linguistic approach characterizing Fuzzy Logic is based on a set of IF-THEN rules, which describe the control behavior and on Membership Functions associated with input and output variables.

Up to 340 Membership Functions, with triangular and trapezoidal shapes, or singleton values are available to describe fuzzy variables.

The Timer/PWM peripheral allows one to manage power devices and timing signals, by implementing different operating modes and high frequency PWM (Pulse Width Modulation) controls. Input Capture and Output Compare functions are available on the Timers.

The Timer has a 16-bit programmable internal Prescaler and a 16-bit Counter, which can use internal or external START/STOP signals and clock.

An internal programmable WATCHDOG is available to avoid loop errors and reset the ICU.

ST52F510/F513/F514 includes a 10-bit, self-calibrating, Analog to Digital Converter with an 8-analog channel Multiplexer. Single/Multiple channels and Single/Sequence conversion modes are supported. External reference can be supplied to obtain more stability and precision in the conversion.

ST52F510/F513/F514 supply different peripherals to implement the most common serial communication protocols. SCI allows the performance of serial asynchronous communication (UART). I<sup>2</sup>C and SPI peripherals allow the implementation of synchronous serial protocols. I<sup>2</sup>C peripherals can work both in master and slave mode. SPI implements Single and Multi Master modes using 3-wire.

Up to 8 interrupt vectors are available, which allow synchronization with peripherals and external devices. Non-Maskable Interrupt and S/W TRAP are available. All interrupts have configurable priority levels and are maskable excluding the Non-Maskable Interrupt, which has fixed top level priority. Two versatile Port Interrupts are available for synchronization with external sources.

The ST52F510/F513/F514 also include an on-chip Power-on-Reset (POR), which provides an internal chip reset during power up situation and a Programmable Low Voltage Detector (PLVD), which causes the ICU to reset if the voltage source V<sub>DD</sub> dips below a threshold. Three programmable thresholds are available, allowing to work with different supply voltages (from 2.7 to 5.5 V).

In order to optimize energy consumption, two different power saving modes are available: Wait mode and Halt mode.

Internal Oscillator at 10 MHz ± 1% is available. External clock, quartz oscillator or RC oscillator are also applicable. The device always starts with the Internal Oscillator, then it reads an Option Byte where the clock mode to be used is programmed.

Program Memory addressing capability addresses up to 8 Kbytes of memory location to store both program instructions and data.

Memory can be locked by the user in order to prevent external undesired operations.

Operations may be performed on data stored in RAM, allowing direct combination of new inputs and feedback data. All RAM bytes are used like Register File.

An additional RAM bench is added to the Program Memory addressing space in order to allow the management of the System/User Stacks and user data storage.

ST52F510/F513/F514 supply the system stack and the user stack located in the additional RAM bench. The user stack can be located anywhere in the additional RAM by writing the top address in the configuration registers, in order to avoid overlap with other data.

Single Voltage Flash allows the user to reprogram the devices on-board by means of the In Situ Programming (ISP) feature. It is possible to store in safe way up to 4K of data in the available EEPROM memory benches. Permanent data, both in Flash and EEPROM can be managed by means of the In-Application-Programming (IAP) feature. Single byte and Page write modes are supported. Flexible write protection, of permanent data or program instructions, is also available.

The Instruction Set composed of up to 107 instructions allows code compression and high speed in the program implementation.

A powerful development environment consisting of a board and software tools allows an easy configuration and use of ST52F510/F513/F514.

The Visual FIVE software tool allows the development and debugging of projects via a user-friendly graphical interface and optimization of generated microcode.

Third-party Hardware Emulators and 'C' Compiler are available to speed-up the application implementation and time-to-market.

### 1.2 Functional Description

ST52F510/F513/F514 ICU's can work in two modes according to the Vpp signal levels:

- Memory Programming Mode
- Working Mode

During Working Mode Vpp must be tied to Vss. To enter the Memory Programming Mode, the Vpp pin must be tied to Vdd.

A RESET signal must be applied to the device to switch from one mode to the other.

#### 1.2.1 Memory Programming Mode.

The ST52F510/F513/F514 memory is loaded in the Memory Programming Mode. All instructions and data are written inside the memory during this phase.

The Option Bytes are loaded during this phase by using the programming tools. The Option Bytes can only be loaded in this phase and cannot be modified run-time.

Data and commands are transmitted by using the I<sup>2</sup>C protocol, implemented using the internal I<sup>2</sup>C peripheral. The In-Situ Programming protocol (ISP) uses the following pins:

- SDA and SCL for transmission
- Vpp for entering in the mode
- RESET for starting the protocol in a stable status
- Vdd and Vss for the power supply.

The Internal clock is used in this phase.

#### 1.2.2 Working Mode.

The processor starts the working phase following the instructions, which have been previously loaded in the first locations of the memory. The first instruction must be a jump to the first program instruction, skipping the data (interrupt vectors, Membership Functions, user data) stored in the first memory page.

ST52F510/F513/F514's internal structure includes two computational blocks, the CONTROL UNIT (CU) and the DATA PROCESSING UNIT (DPU), which performs boolean functions. The DECISION PROCESSOR (DP) block cooperates with these blocks to perform Fuzzy algorithms.

The DP can manage up to 340 different Membership Functions for the antecedent part of fuzzy rules. The consequent terms of the rules are "crisp" values (real numbers). The maximum number of rules that can be defined is limited by the dimensions of the standard algorithm implemented.

The Program/Data Memory is shared between Fuzzy and standard algorithms. Within this memory, the user data can be stored both in non volatile memory as well as in the RAM locations.

The Control Unit (CU) reads information and the status of the peripherals.

Arithmetic calculus can be performed on these values by using the internal CU and Register File, which supports all computations. The peripheral inputs can be Fuzzy and/or arithmetic output values contained in the Register File or Program/Data Memory.



Table 1.1 ST52F510/F513/F514 Devices Summary

Device	NVM	RF	RAM	EEPROM	Timers	ADC	Comms	I/O	Package
ST52F510YmM6	4/8 K FLASH	256	256	-	2X16-bit	10-bit 2 Ch	SCI I <sup>2</sup> C	10	So 16
ST52F510FmM6	4/8 K FLASH	256	256	-	2X16-bit	10-bit 6 Ch	SCI I <sup>2</sup> C	14	So 20
ST52F510FmB6	4/8 K FLASH	256	256	-	2X16-bit	10-bit 6 Ch	SCI I <sup>2</sup> C	14	Dip 20
ST52F510GmM6	4/8 K FLASH	256	256	-	2X16-bit	10-bit 8 Ch	SCI I <sup>2</sup> C SPI	22	So 28
ST52F510GmB6	4/8 K FLASH	256	256	-	2X16-bit	10-bit 8 Ch	SCI I <sup>2</sup> C SPI	22	SDip 32
ST52F513YmM6	4/8 K FLASH	256	256	256	2X16-bit	10-bit 2 Ch	SCI I <sup>2</sup> C	10	So 16
ST52F513FmM6	4/8 K FLASH	256	256	256	2X16-bit	10-bit 6 Ch	SCI I <sup>2</sup> C	14	So 20
ST52F513FmB6	4/8 K FLASH	256	256	256	2X16-bit	10-bit 6 Ch	SCI I <sup>2</sup> C	14	Dip 20
ST52F513GmM6	4/8 K FLASH	256	256	256	2X16-bit	10-bit 8 Ch	SCI I <sup>2</sup> C SPI	22	So 28
ST52F513GmB6	4/8 K FLASH	256	256	256	2X16-bit	10-bit 8 Ch	SCI I <sup>2</sup> C SPI	22	SDip 32
ST52F514YmM6	4 K FLASH	256	256	1024 / 4096	2X16-bit	10-bit 2 Ch	SCI I <sup>2</sup> C	10	So 16
ST52F514FmM6	4 K FLASH	256	256	1024 / 4096	2X16-bit	10-bit 6 Ch	SCI I <sup>2</sup> C	14	So 20
ST52F514FmB6	4 K FLASH	256	256	1024 / 4096	2X16-bit	10-bit 6 Ch	SCI I <sup>2</sup> C	14	Dip 20
ST52F514GmM6	4 K FLASH	256	256	1024 / 4096	2X16-bit	10-bit 8 Ch	SCI I <sup>2</sup> C SPI	22	So 28
ST52F514GmB6	4 K FLASH	256	256	1024 / 4096	2X16-bit	10-bit 8 Ch	SCI I <sup>2</sup> C SPI	22	SDip 32

COMMON FEATURES	ST52F510/F513/F514
Watchdog	Yes
Other Features	NMI, PLVD, POR
Temperature Range	From -40° to +85°
Operating Supply	2.7 - 5.5 V
CPU Frequency	from 1 to 24 MHz.

**Legend:**

<b>Sales code:</b>	<b>ST52tnnnncmpy</b>
<b>Memory type (t):</b>	F=FLASH
<b>Subfamily (nnn):</b>	510, 513, 514
<b>Pin Count (c):</b>	Y=16 pins, F=20 pins, G=28 pins, K=32/34 pins
<b>Memory Size (m):</b>	2=4 Kb, 3=8 Kb Flash (ST52F510 & ST52F513) 1=1024, 3=4096 EEPROM (only ST52F514)
<b>Packages (p):</b>	B=PDIP, M=PSO, T=TQFP
<b>Temperature (y):</b>	0=+25, 1=0 +70, 3=-40 +125, 5=-10 +85, 6=-40 +85, 7=-40 +105

Figure 1.1 ST52F510/F513/F514 Block Diagram

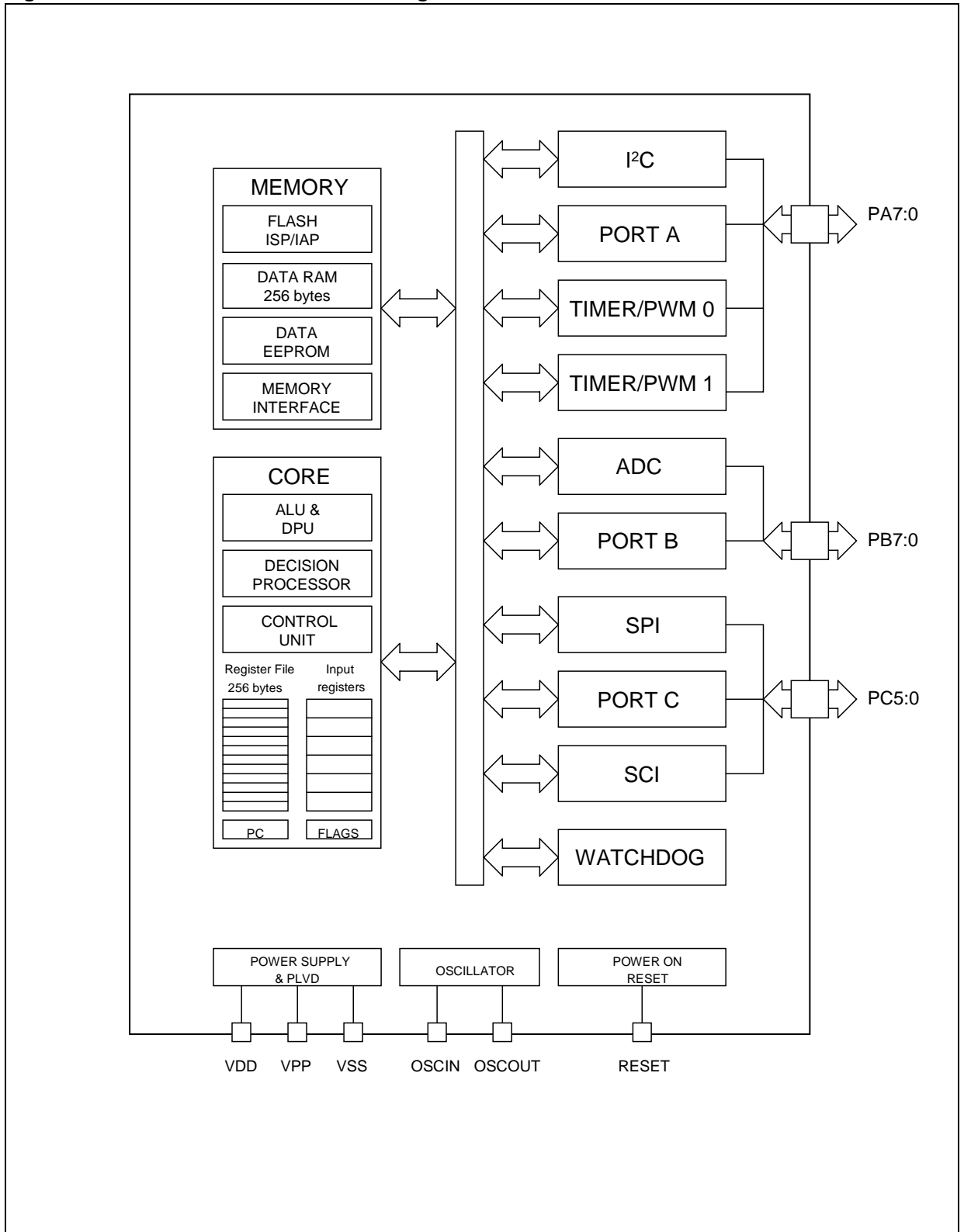


Figure 1.2 ST52F510/F513/F514 SO20/DIP20 Pin Configuration

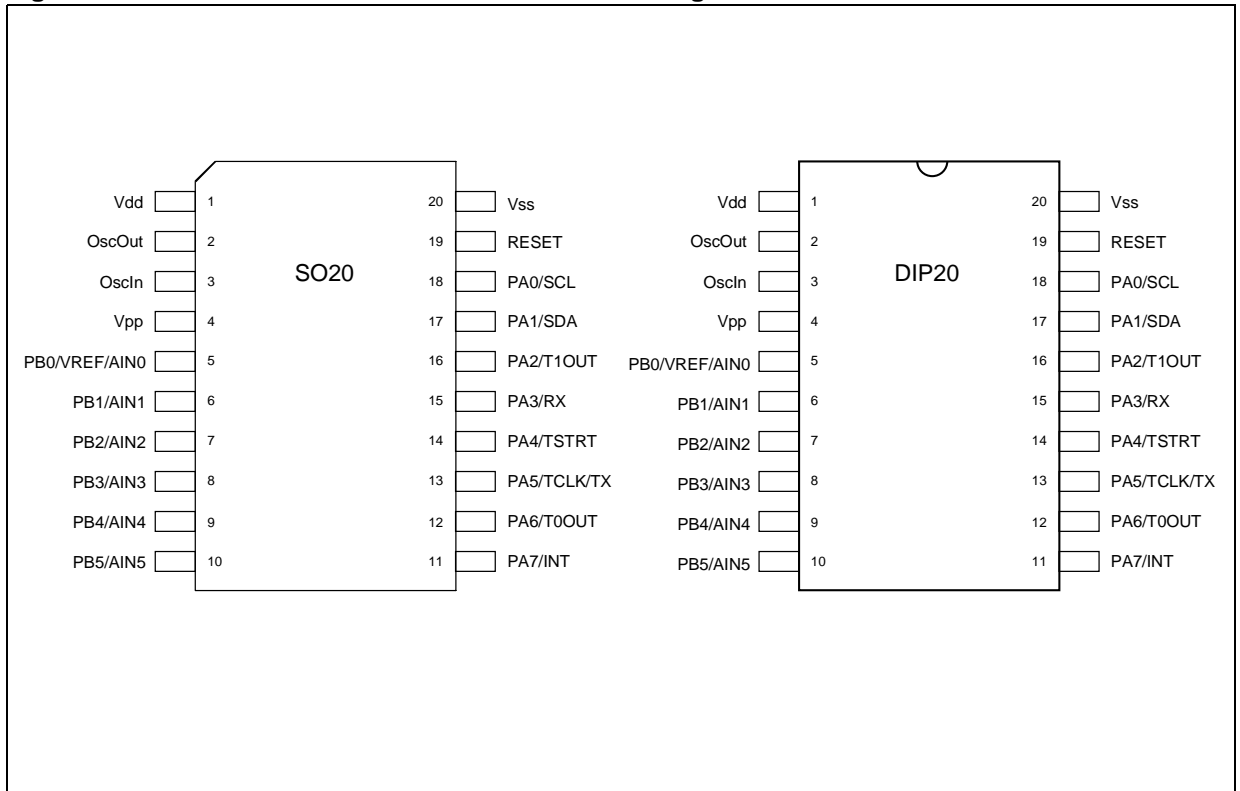


Figure 1.3 ST52F510/F513/F514 SO16 Pin Configuration

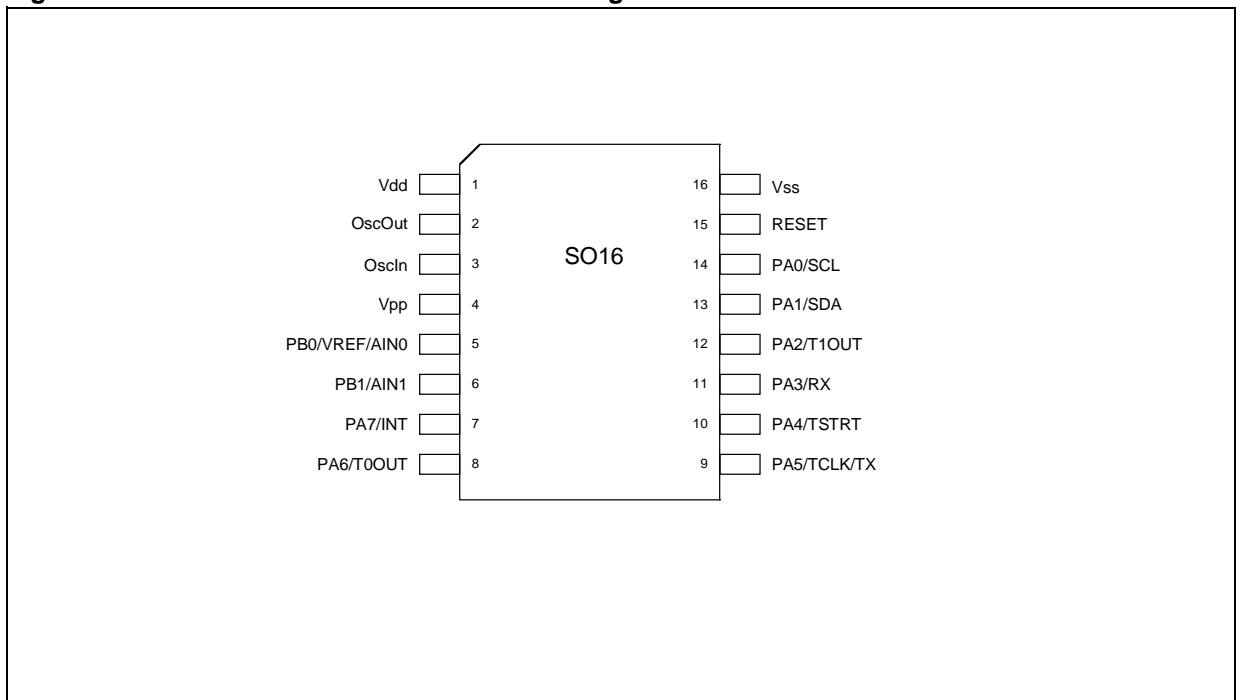


Figure 1.4 ST52F510/F513/F514 SDIP32/DIP28 Pin Configuration

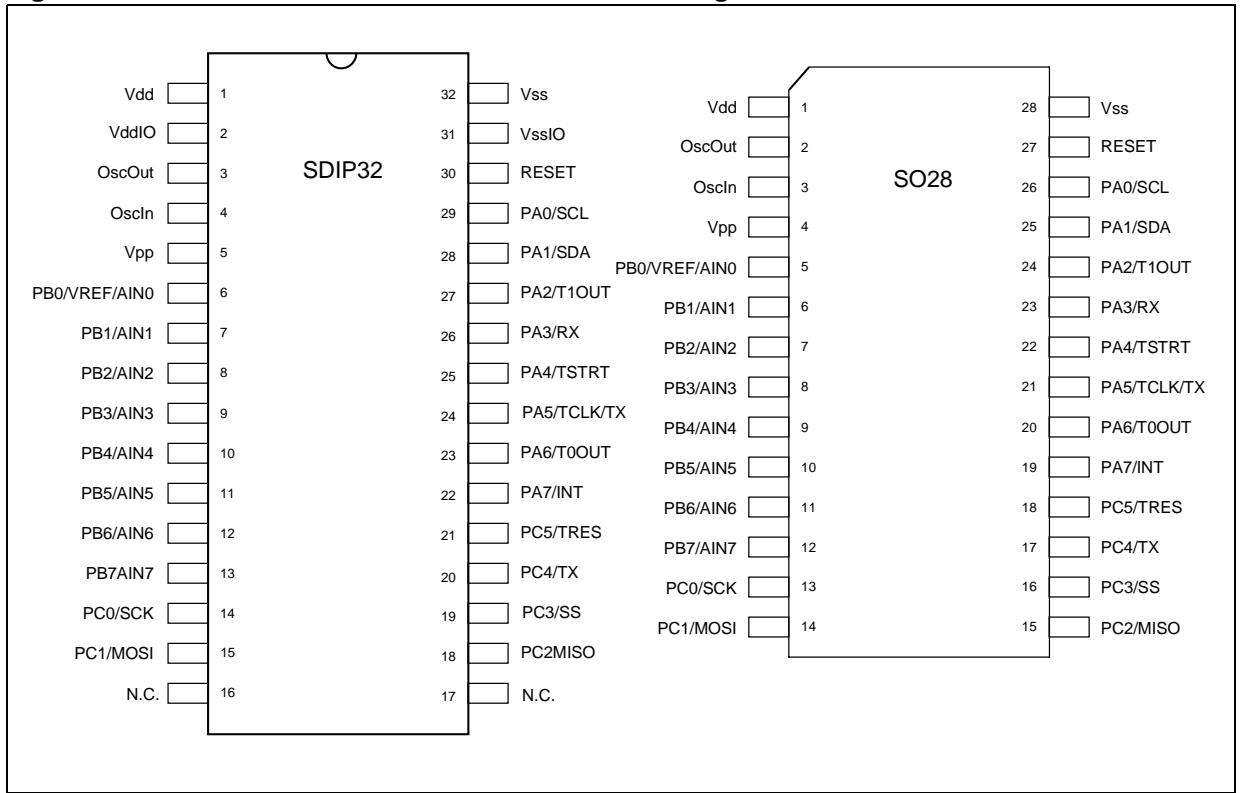


Table 1.2 ST52F510/F513/F514 SDIP32 Pin List

SDIP32	NAME	Programming Phase	Working Phase
1	Vdd	Digital Power Supply	Digital Power Supply
2	VddIO	Digital Power Supply	Digital I/O Ports Power Supply
3	OSCOOUT		Oscillator Output
4	OSCIN		Oscillator Input
5	Vpp	Programming Mode Selector	Programming Mode Selector
6	PB0/VREF/AIN0		Digital I/O, A/D Voltage Reference, Analog Input
7	PB1/AIN1		Digital I/O, Analog Input
8	PB2/AIN2		Digital I/O, Analog Input
9	PB3/AIN3		Digital I/O, Analog Input
10	PB4/AIN4		Digital I/O, Analog Input
11	PB5/AIN5		Digital I/O, Analog Input
12	PB6/AIN6		Digital I/O, Analog Input
13	PB7/AIN7		Digital I/O, Analog Input
14	PC0/SCK		Digital I/O, SPI Serial Clock
15	PC1/MOSI		Digital I/O, SPI Master out Slave in
16	N.C		Not Connected
17	N.C		Not Connected
18	PC2/MISO		Digital I/O, SPI Master in Slave out
19	PC3/SS		Digital I/O, SPI Slave Select
20	PC4/TX		Digital I/O, SCI Transmission
21	PC5/TRES		Digital I/O, Timer/PWM 0 Reset
22	PA7/INT		Digital I/O, Non Maskable Interrupt
23	PA6/T0OUT		Digital I/O, Timer/PWM 0 output
24	PA5/TCLK/TX		Digital I/O, Timer/PWM 0 clock
25	PA4/TSTRT		Digital I/O, Timer/PWM 0 start/stop
26	PA3/RX		Digital I/O, SCI Reception
27	PA2/T1OUT		Digital I/O, Timer/PWM 1 output
28	PA1/SDA	Serial Data I/O	Digital I/O, I <sup>2</sup> C Serial Data I/O
29	PA0/SCL	Serial Clock	Digital I/O, I <sup>2</sup> C Serial Clock
30	RESET	General Reset	General Reset
31	VssIO	Digital Ground	Digital I/O Ports Ground
32	Vss	Digital Ground	Digital Ground

**Table 1.3 ST52F510/F513/F514 SO28 Pin List**

SO28	NAME	Programming Phase	Working Phase
1	Vdd	Digital Power Supply	Digital Power Supply
2	OSCOOUT		Oscillator Output
3	OSCIN		Oscillator Input
4	Vpp	Programming Mode Selector	Programming Mode Selector
5	PB0/VREF/AIN0		Digital I/O, A/D Voltage Reference, Analog Input
6	PB1/AIN1		Digital I/O, Analog Input
7	PB2/AIN2		Digital I/O, Analog Input
8	PB3/AIN3		Digital I/O, Analog Input
9	PB4/AIN4		Digital I/O, Analog Input
10	PB5/AIN5		Digital I/O, Analog Input
11	PB6/AIN6		Digital I/O, Analog Input
12	PB7/AIN7		Digital I/O, Analog Input
13	PC0/SCK		Digital I/O, SPI Serial Clock
14	PC1/MOSI		Digital I/O, SPI Master out Slave in
15	PC2/MISO		Digital I/O, SPI Master in Slave out
16	PC3/SS		Digital I/O, SPI Slave Select
17	PC4/TX		Digital I/O, SCI Transmission
18	PC5/TRES		Digital I/O, Timer/PWM 0 Reset
19	PA7/INT		Digital I/O, Non Maskable Interrupt
20	PA6/T0OUT		Digital I/O, Timer/PWM 0 output
21	PA5/TCLK/TX		Digital I/O, Timer/PWM 0 clock
22	PA4/TSTRT		Digital I/O, Timer/PWM 0 start/stop
23	PA3/RX		Digital I/O, SCI Reception
24	PA2/T1OUT		Digital I/O, Timer/PWM 1 output
25	PA1/SDA	Serial Data I/O	Digital I/O, I <sup>2</sup> C Serial Data I/O
26	PA0/SCL	Serial Clock	Digital I/O, I <sup>2</sup> C Serial Clock
27	RESET	General Reset	General Reset
28	Vss	Digital Ground	Digital Ground

Table 1.4 ST52F510/F513/F514 SO20/DIP20/SO16 Pin List

SO20 DIP20	SO16	NAME	Programming Phase	Working Phase
1	1	Vdd	Digital Power Supply	Digital Power Supply
2	2	OSCOOUT		Oscillator Output
3	3	OSCCIN		Oscillator Input
4	4	Vpp	Programming Mode Selector	Programming Mode Selector
5	5	PB0/VREF/AIN0		Digital I/O, A/D Voltage Reference, Analog Input
6	6	PB1/AIN1		Digital I/O, Analog Input
7	-	PB2/AIN2		Digital I/O, Analog Input
8	-	PB3/AIN3		Digital I/O, Analog Input
9	-	PB4/AIN4		Digital I/O, Analog Input
10	-	PB5/AIN5		Digital I/O, Analog Input
11	7	PA7/INT		Digital I/O, Non Maskable Interrupt
12	8	PA6/T0OUT		Digital I/O, Timer/PWM 0 output
13	9	PA5/TCLK/TX		Digital I/O, Timer/PWM 0 clock, SCI transmission
14	10	PA4/TSTRT		Digital I/O, Timer/PWM 0 start/stop
15	11	PA3/RX		Digital I/O, SCI Reception
16	12	PA2/T1OUT		Digital I/O, Timer/PWM 1 output
17	13	PA1/SDA	Serial Data I/O	Digital I/O, I <sup>2</sup> C Serial Data I/O
18	14	PA0/SCL	Serial Clock	Digital I/O, I <sup>2</sup> C Serial Clock
19	15	RESET	General Reset	General Reset
20	16	Vss	Digital Ground	Digital Ground

### 1.3 Pin Description

ST52F510/F513/F514 pins can be set in digital input mode, digital output mode, interrupt mode or in Alternate Functions. Pin configuration is achieved by means of the configuration registers. The functions of the ST52F510/F513/F514 pins are described below:

**V<sub>DD</sub>**. Main Power Supply Voltage.

**V<sub>DDIO</sub>**. I/O Ports Power Supply Voltage. It is recommended to connect this pin with a supply voltage de-coupled with V<sub>DD</sub> in order to improve the immunity from the noise generated by the I/O switching.

**V<sub>SS</sub>**. Digital circuit Ground.

**V<sub>SSIO</sub>**. I/O Ports Ground. See V<sub>DDIO</sub>

**V<sub>PP</sub>**. Programming/Working mode selector. During the Programming phase V<sub>PP</sub> must be set to V<sub>DD</sub>. In Working phase V<sub>PP</sub> must be equal to V<sub>SS</sub>.

**OSCin** and **OSCout**. These pins are internally connected to the on-chip oscillator circuit. A quartz crystal or a ceramic resonator can be connected between these two pins in order to allow correct use of ST52F510/F513/F514 with various stability/cost trade-offs. An external clock signal can be applied to OSCin: in this case OSCout must be grounded. To reduce costs, an RC circuit can be applied to the OSCin pin to establish the internal clock frequency, instead of the quartz. Without any connection, the device can work with its internal clock generator (10 MHz)

**RESET**. This signal is used to reset the ST52F510/F513/F514 and re-initialize the registers and control signals. It is also used when switching from the Programming Mode to Working Mode and vice versa.

**PA0-PA7, PB0-PB7, PC0-PC5**. These lines are organized as I/O ports. Each pin can be configured as an input, output (with pull-up, push-pull, weak-pull-up, open-drain, high-impedance), or as an interrupt source.

**VREF, AIN0-AIN7**. These pins are used to input the analog signals into the A/D Converter. An analog multiplexer is available to switch these inputs to the A/D Converter. The pin VREF is used to input an external A/D Reference Voltage.

**T0OUT, T1OUT**. These pins output the signals generated by the PWM/Timer 0 and PWM/Timer 1 peripheral.

**TRES, TSTRT, TCLK**. These pins are related to the PWM/Timer 0 peripheral and are used for Input Capture and event counting. The TRES pin is used to set/reset the Timer; the TSTRT pin is used to start/stop the counter. The Timer can be driven by the internal clock or by an external signal connected to the TCLK pin.

**INT**. This pin is used as input for the Non-Maskable (top level) interrupt. The interrupt signal is detected only if the pin is configured in Alternate Function.

**SCL, SDA**. These pins are used respectively as Serial Clock and Serial Data I/O in I<sup>2</sup>C peripheral protocol. They are used also in Programming Mode to receive and transmit data.

**TX, RX**. Serial data output of SCI Transmitter block (TX) and Serial data input of the SCI Receiver block (RX).

**SCK, MISO, MOSI, SS**. These pins are used by the Serial Peripheral Interface (SPI) peripheral. SCK is the serial clock line. MISO (Master In Slave Out) and MOSI (Master Out Slave In) are the serial data lines, which work in input or in output depending on if the device is working in slave or master mode. The SS pin allows the selection of the device master/slave mode.



## 2 INTERNAL ARCHITECTURE

ST52F510/F513/F514's architecture is Register File based and is composed of the following blocks and peripherals:

- Control Unit (CU)
- Data Processing Unit (DPU)
- Decision Processor (DP)
- ALU
- Memory Interface
- up to 256 bytes Register File
- Program/Data Memory
- Data EEPROM
- Interrupts Controller
- Clock Oscillator
- PLVD and POR
- Digital I/O ports
- Analog Multiplexer and A/D Converter
- Timer/PWMs
- I<sup>2</sup>C
- SPI
- SCI

### 2.1 Control Unit and Data Processing Unit

The Control Unit (CU) decodes the instructions stored in the Program Memory and generates the appropriate control signals. The main parts of the CU are illustrated in Figure 2.1.

The five different parts of the CU manage Loading, Logic/Arithmetic, Jump, Control and the Fuzzy instruction set.

The block called "Collector" manages the signals deriving from the different parts of the CU. The collector defines the signals for the Data Processing Unit (DPU) and Decision Processor (DP), as well as for the different peripherals of the ICU.

The block called "Arbiter" manages the different parts of the CU, so that only one part of the system is activated during working mode.

The CU structure is extremely flexible and was designed with the purpose of easily adapting the core of the microcontroller to market needs. New instruction sets or new peripherals can easily be included without changing the structure of the microcontroller, maintaining code compatibility.

A set of 107 different instructions is available. Each instruction requires a number of clock pulses to be performed that depends on the complexity of the instruction itself. The clock pulses to execute the instructions are driven directly by the masterclock, which has the same frequency of the oscillator signal supplied.

Figure 2.1 CU Block Diagram

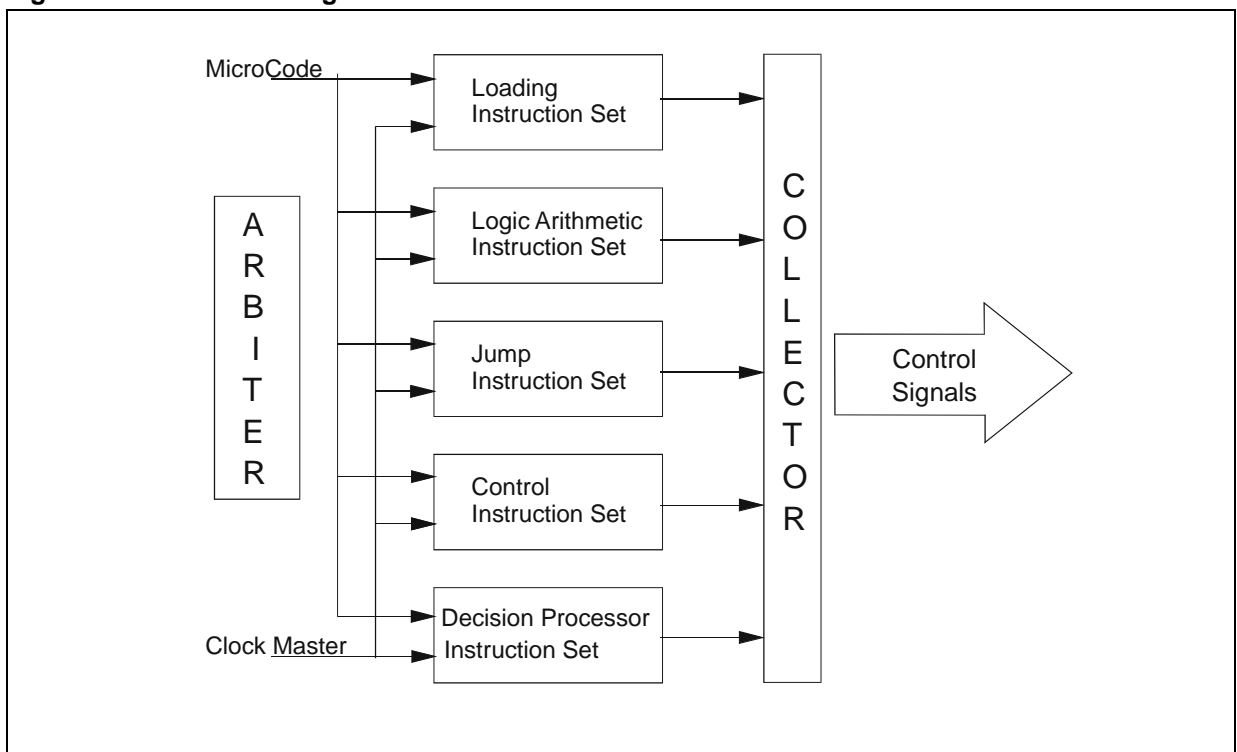
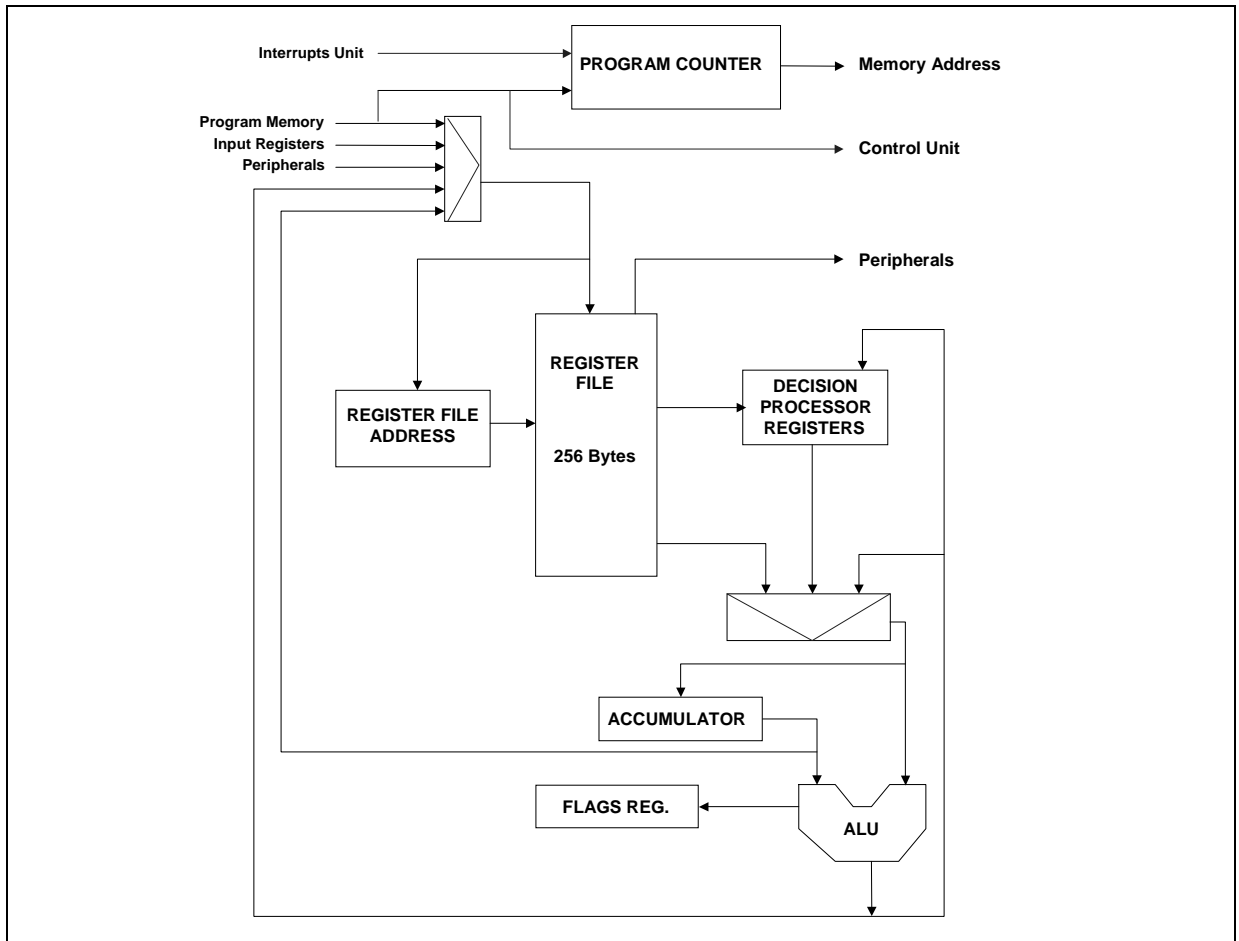


Figure 2.2 Data Processing Unit (DPU)



The DPU receives, stores and sends the instructions deriving from the Program/Data Memory, Register File or from the peripherals. It is controlled by the CU on the basis of the decoded instruction. The Fuzzy registers store the partial results of the fuzzy computation. The accumulator register is used by the ALU and is not accessible directly: the instructions used by the ALU can address all the Register File locations as operands, allowing a more compact code and a faster execution.

The following addressing modes are available: inherent, immediate, direct, indirect, bit direct.

**2.1.1 Program Counter.**

The Program Counter (PC) is a 16-bit register that contains the address of the next memory location to be processed by the core. This memory location may be both an instruction or data address.

The Program Counter's 16-bit length allows the direct addressing of a maximum of 64 Kbytes in the Program/Data Memory space.

The PC can be changed in the following ways:

- JP (Jump)                    PC = Jump Address
- Interrupt                    PC = Interrupt Vector
- RETI                         PC = Pop (stack)
- RET                         PC = Pop (stack)
- CALL                        PC = Subroutines address
- Reset                        PC = Reset Vector
- Normal Instruction    PC = PC + 1

**2.1.2 Flags.**

The ST FIVE core includes different sets of flags that correspond to 2 different modes: normal mode and interrupt mode. Each set of flags consist of a CARRY flag (C), ZERO flag (Z) and SIGN flag (S). Each set is stacked: one set of flags is used during normal operation and other sets are used during each level of interrupt. Formally, the user has to manage only one set of flags: C, Z and S since the flag stack operation is performed automatically.

Each interrupt level has its own set of flags, which is saved in the Flag Stack during interrupt servicing. These flags are restored from the Flag Stack automatically when a RETI instruction is executed.

If the ICU was in normal mode before an interrupt, after the RETI instruction is executed, the normal flags are restored.

**Note:** A subroutine CALL is a normal mode execution. For this reason a RET instruction, consequent to a CALL instruction, doesn't affect the normal mode set of flags.

Flags are not cleared during context switching and remain in the state they were in at the exit of the last interrupt routine switching.

The Carry flag is set when an overflow occurs during arithmetic operations, otherwise it is cleared. The Sign flag is set when an underflow occurs during arithmetic operations, otherwise it is cleared.

The flags, related to the current context, can be checked by reading the FLAGS Input Register 38 (026h).

## 2.2 Arithmetic Logic Unit

The 8-bit Arithmetic Logic Unit (ALU) performs arithmetic calculations and logic instructions such as: sum, subtraction, bitwise AND, OR, XOR, bit set and reset, bit test and branch, right/left shift and rotate (see the Chapter 9 Instruction Set for further details).

In addition, the ALU of ST52F510/F513/F514 can perform multiplication (MULT) and division (DIV). Multiplication is performed by using 8 bit operands storing the result in 2 registers (16 bit values); the division instruction addresses the MSB of the dividend (the LSB is stored in the next address): the result and remainder are stored in these source addresses (see Figure 2.3 and Figure 2.4).

In order to manage signed type values, the ALU also performs addition and subtraction with offset (ADDO and SUBO). These instructions respectively subtract and add 128 to the overall result, in order to manage values logically in the range between -128,127.

Figure 2.3 Multiplication

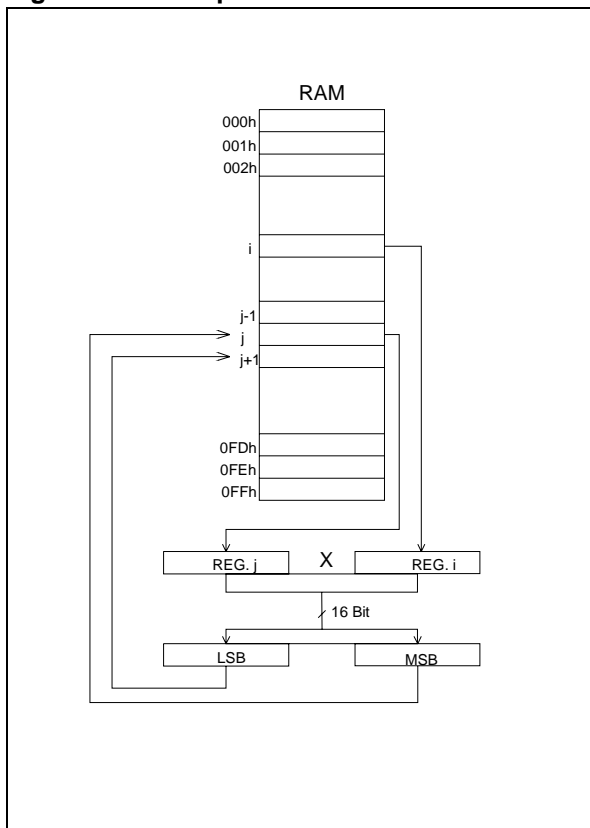
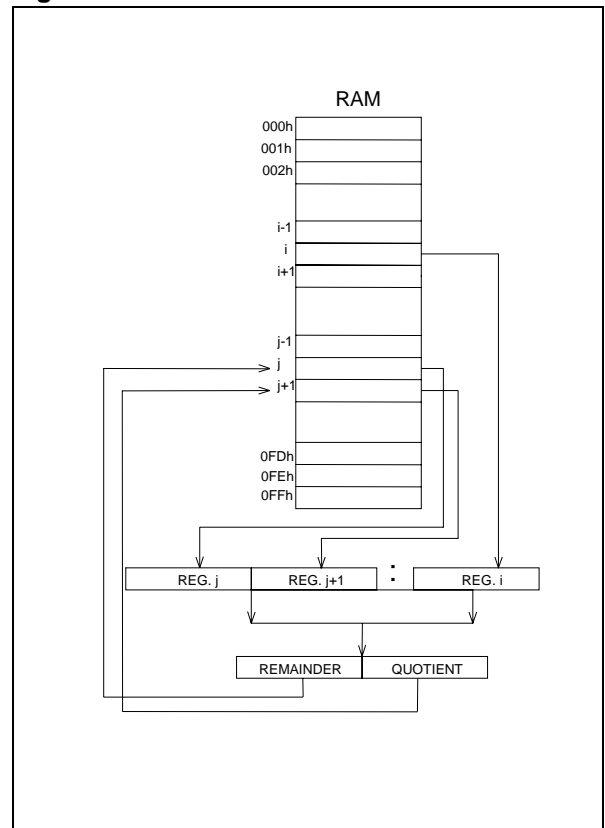


Figure 2.4 Division



**2.3 Register Description****Flags Register (FLAG)**

Input Register 38 (026h) Read Only

Reset Value: 0000 0000 (00h)

7							0
-	-	-	-	-	Z	S	C

Bit 7-3: Not Used

Bit 2: **Z** Zero flagBit 1: **S** Sign flagBit 0: **C** Carry flag

### 3 ADDRESSING SPACES

ST52F510/F513/F514 has six separate addressing spaces:

- Register File
- Program/Data Memory
- Stacks
- Input Registers
- Output Registers
- Configuration Registers

Each space is addressed by a load type instruction that indicates the source and the destination space in the mnemonic code (see Figure 3.1).

#### 3.1 Memory Interface

The read/write operation in the space addresses are managed by the Memory Interface, which can recognize the type of memory addressed and set the appropriate access time and mode.

In addition, the Memory Interface manages the In Application Programming (IAP) functions in Flash devices like writing cycle and memory write protection.

#### 3.2 Register File

The Register File consists of 256 general purpose 8-bit RAM locations called “registers” in order to recall the functionality.

The Register File exchanges data with all the other addressing spaces and is used by the ALU to perform all the arithmetic and logic instructions. These instructions have any Register File address as operands.

Data can be moved from one location to another by using the LDRR instruction; see further ahead for information on the instruction used to move data between the Register File and the other addressing spaces.

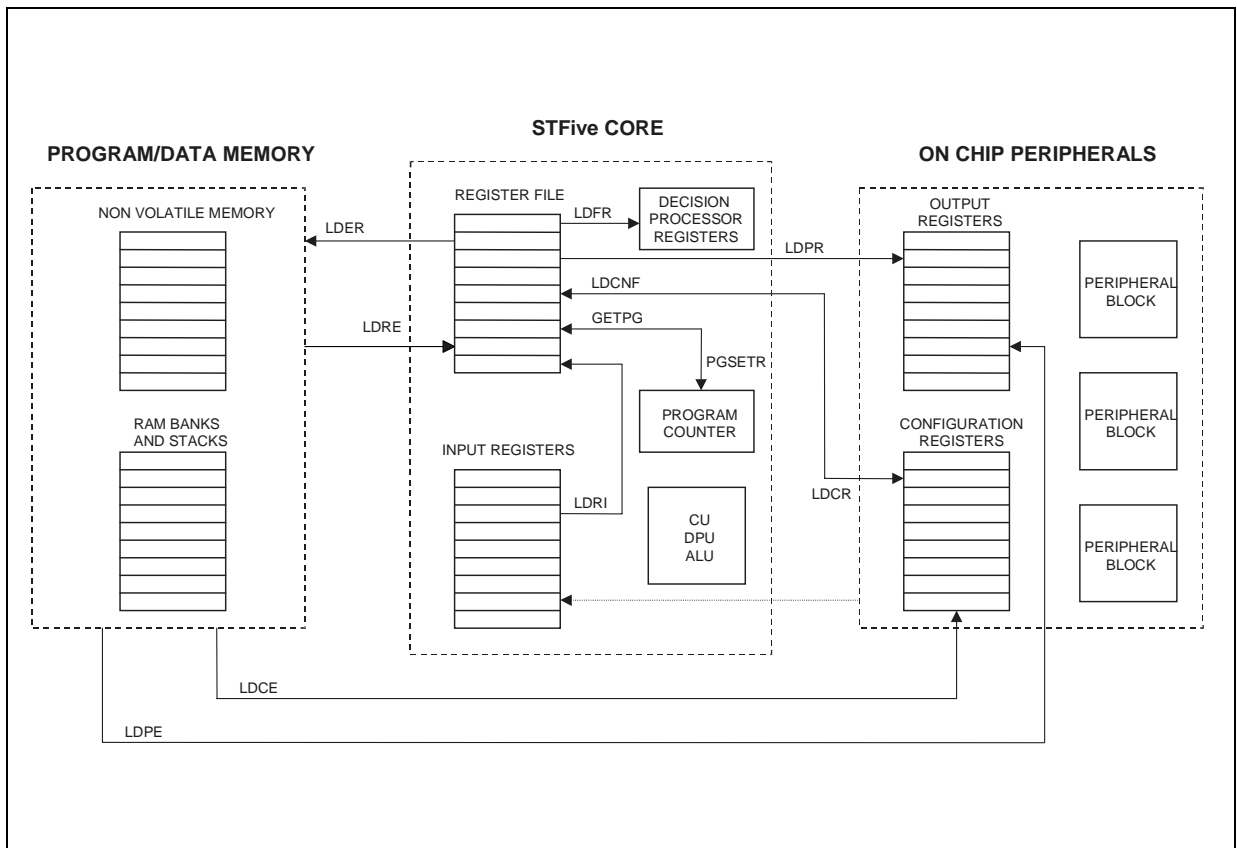
#### 3.3 Program/Data Memory

The Program/Data Memory consists of both non-volatile memory (Flash, EEPROM) and RAM memory benches.

Non-volatile memory (NVM) is mainly used to store the user program and can also be used to store permanent data (constant, look-up tables).

Each RAM bench consists of 256 locations used to store run-time user data. At least one bench is present in the devices. RAM benches are also used to implement both System and User Stacks.

Figure 3.1 Addressing Spaces



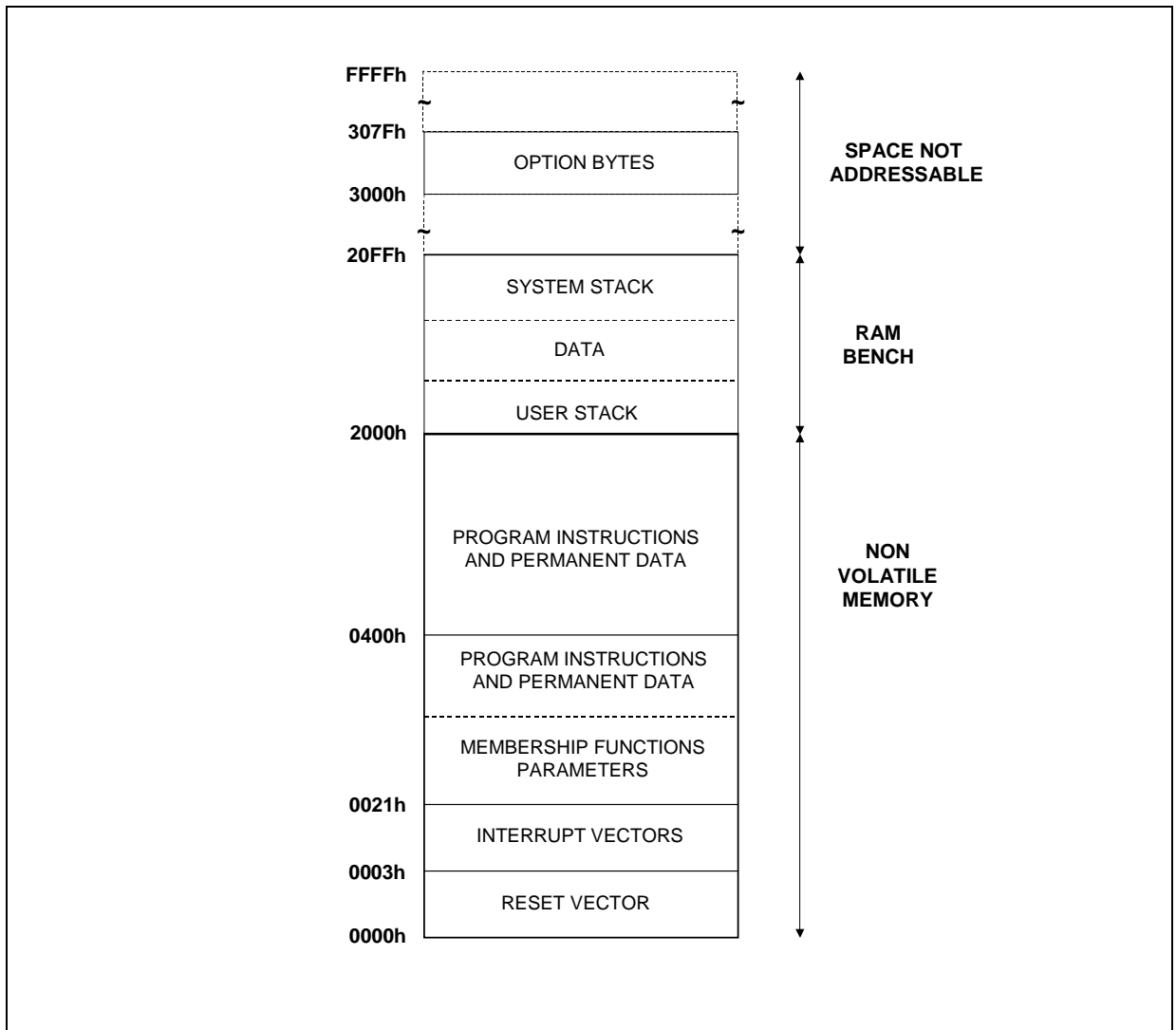
NVM is always located beginning after the first locations of the addressing space. RAM banks are always located after NVM.

NVM is organized in accordance to the following blocks (see Figure 3.2):

- *Reset Vector block* (from address 0 to 2) contains an absolute jump instruction to the first user program instruction. The Assembler tool automatically fills these locations with correct data.
- *Interrupt Vectors block* (from location 3 up to 32) contains the interrupt vectors. Each address is composed of three bytes (the jump opcode and the 16 bit address). Interrupt vectors are set by using IRQ pseudo-instruction (see the Programming Manual).

- *Mbfs Setting block* (just after the interrupt vectors) contains the coordinates of the vertexes of every Mbf defined in the program. The last address that can be assigned to this block is 1023. This area is dynamically assigned according to the size of the fuzzy routines. The memory area that remains unused, if any, is assigned to the Program Instructions block.
- *The Program Instructions block* (just after the last Mbf data through the last NVM address) contains the instruction of the user program and the permanent data.
- *Option bytes block* (from location 3000h to 307Fh) is the addressing space reserved for the option bytes. In ST52F510/F513/F514, only the location from 3000h to 3007h are used.

Figure 3.2 Program/Data Memory Organization



Flash and EEPROM are programmed electrically just applying the supply voltage (2.7 V to 5.5 V) and it is also erased electrically; this feature allows the user to easily reprogram the memory without taking the device off from the board (In Situ Programming ISP). Data and commands are transmitted through the I<sup>2</sup>C serial communication protocol. Data can also be written run-time with the In Application Programming (IAP)

NVM can be locked by the user during the programming phase, in order to prevent external operation such as reading the program code and assuring protection of user intellectual property. Flash and EEPROM pages can be protected by unintentional writings.

The operations that can be performed on the NVM during the Programming Phase, ISP and IAP are described in detail in the Section 4.

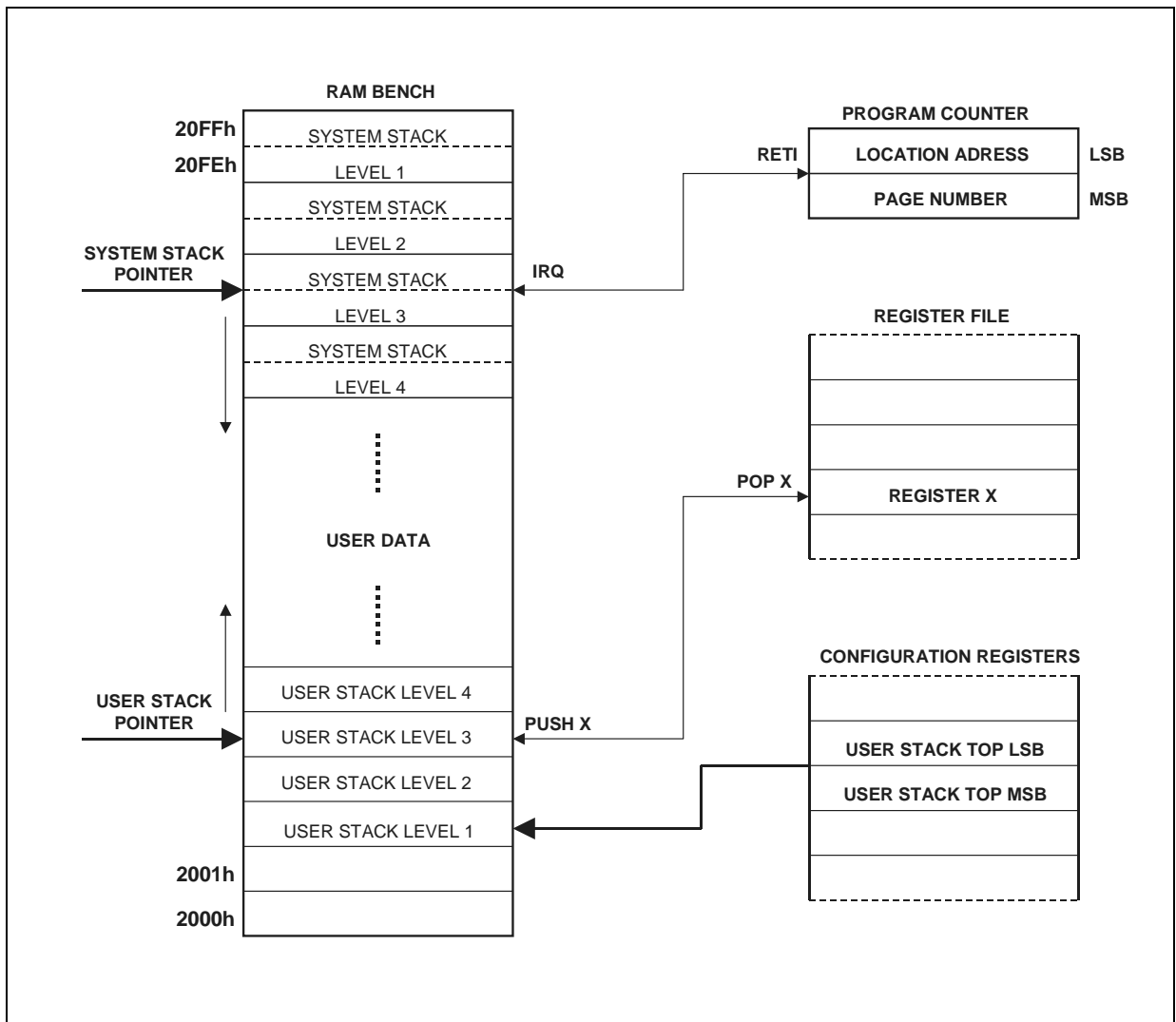
### 3.4 System and User Stacks

The System and User Stacks are located in the Program/Data memory in the RAM benches.

System Stacks are used to push the Program Counter (PC) after an Interrupt Request or a Subroutine Call. After a RET (Return from a subroutine) or a RETI (Return from an interrupt) the PC that is saved is popped from the stack and restored. After an interrupt request, the flags are also saved in a reserved stack inside the core, so each interrupt has its own flags.

The System Stack is located in the last RAM bench starting from the last address (255) inside the bench page. The System Stack Pointer (SSP) can be read and modified by the user. For each level of stack 2 bytes of the RAM are used. The SSP points to the first currently available stack position. When a subroutine call or interrupt request occurs, the content of the PC is stored in a couple of locations pointed to by the SSP that is decreased by 2.

Figure 3.3 System and User Stack



When a return occurs (RET or RETI instruction), the SSP is increased by 2 and the data stored in the pointed locations couple is restored back into the PC.

The current SSP can be read and write in the couple of Configuration Registers 44 02Ch (MSB: page number, always 32 020h) and 45 02Dh (LSB: location address) (see Figure 3.3). In ST52F510/F513/F514 the user can only consider the LSB because the MSB is always the same.

The User Stack is used to store user data and is located beginning from a RAM bench location set by the user (USTP) by writing the couple of Configuration Registers 5 005h (MSB: page number) and 6 005h (LSB: location address) (see Figure 3.3). Register 5, which is the page number, must always be set to a value between 32 (020h) and 255 (0FFh): values higher than 32 always address RAM on page 32.

**Note:** In ST52F510/F513/F514 MSB doesn't have to be set or read because the RAM is only 256 bytes. The LSB of the user stack is equal to 0 at reset. The LSB of the system stack is equal to 255 at reset.

This feature allows a flexible use of the User Stack in terms of dimension and to avoid overlaps. The User Stack Pointer (USP) points to the first currently available stack location. When the user stores a byte value contained in the Register File by using the PUSH instruction, the value is stored in the position pointed to by the USP that is increased (the User Stack order is opposite to the System Stack one). When the user takes a value from the User Stack with the POP instruction, the USP is decreased and the value pointed to is copied in the specified Register File location.

By writing the USTP, the new address is automatically written in the USP. The current USP can be read from the Input Registers 11 0Bh (MSB: page number, always 32 020h) and 12 0Ch (LSB: location address) (see Figure 3.3). In ST52F510/F513/F514 the user can only consider the LSB because the MSB is always the same.

**Note:** The user must pay close attention to avoid overlapping user and Stacks data. The User Stack Top location and the System Stack Pointer should be configured with care in order to have enough space between the two stacks.

### 3.5 Input Registers

The ST52F510/F513/F514 Input Registers bench consists of a file of 8-bit registers containing data or the status of the peripherals. For example, the

Input Registers contain data converted by the ADC, Ports, serial communication peripherals, Timers, etc.

The Input Registers can be accessed by using the LDRI instruction that loads the specified Register File address with the contents of the specified Input Register. See the Programming Manual for further details on this instruction. The Input Registers are read-only registers.

In order to simplify the concept, a mnemonic name is assigned to each register. The same name is used in Visual FIVE development tools. The list of the Input Registers is shown in Table 3.1.

### 3.6 Output registers

The ST52F510/F513/F514 Output Registers bench consists of a file of 8-bit registers containing data sent to the Peripherals and the I/O Ports (for example: Timer Counters, data to be transmitted by the serial communication peripherals, data to be sent to the Port pins in output, etc.).

The registers are located inside the Peripherals and Ports, which allow flexibility and modularity in the design of new family devices.

The Output Registers are write only. In order to access the configuration Register the user can use the following instructions:

- LDPI: loads the immediate value in the specified Output Register.
- LDPR: loads the contents of the specified Register File location into the output register specified. This instruction allows computed data to be sent to Peripherals and Ports.
- LDPE direct: loads the contents of the specified Program/Data Memory location into the output register specified. This instruction allows data to be sent to Peripherals and Ports from a table.
- LDPE indirect: loads the contents of the Program/Data Memory location whose address is contained in the specified Register File location into the output register specified. This instruction allows data to be sent to Peripherals and Ports from a table pointed to by a register.

See the Programming manual for further details about these instructions.

In order to simplify the concept, a mnemonic name is assigned to each register. The same name is used in Visual FIVE development tools. The list of the Output Registers is shown in Table 3.2.



### 3.7 Configuration Registers & Option Bytes

The ST52F510/F513/F514 Configuration Registers bench consists of a file of 8-bit registers that allows the configuration of all the ICU blocks. The registers are located inside the block they configure in order to obtain greater flexibility and modularity in the design of new family devices. In the Configuration Registers, each bit has a peculiar use, so the logic level of each of them must be considered.

Some special configuration data, that needs to be load at the start-up and not further changed, are stored in Option Bytes. These are loaded only during the device programming phase. See Table 3.3 and Section 4 for a detailed description of the Option Bytes.

The Configuration Registers are readable and writable; the addresses refer to the same register both in read and in write. In order to access the Configuration Register the user can work in several modes by utilizing the following instructions:

- LDCI: loads the immediate value in the Configuration Register specified and is the most commonly used to write configuration data.
- LDCR: loads the Configuration Register specified with the contents of the specified Register File location, allowing a parametric configuration.
- LDCE: loads the Configuration Register specified with the contents of the specified Program/Data Memory location, allowing the configuration data to be taken from a table.
- LDCNF: loads the Register File location specified with the contents of the Configuration Register indicated, allowing for the inspection of the configuration of the device (permitting safe run-time modifications).

In order to simplify the concept, a mnemonic name is assigned to each register. The same name is used in Visual FIVE development tools. The list of the Configuration Registers is shown in Table 3.4.

Table 3.1 Input Registers

Mnemonic	Description	Address	
PORT_A_IN	Port A data Input Register	0	00h
PORT_B_IN	Port B data Input Register	1	01h
PORT_C_IN	Port C data Input Register	2	02h
-	Not Used	3	03h
-	Not Used	4	04h
SPI_IN	Serial Peripheral Interface data Input Register	5	05h
I2C_IN	I <sup>2</sup> C Interface data Input Register	6	06h
I2C_SR1	I <sup>2</sup> C Interface Status Register 1	7	07h
I2C_SR2	I <sup>2</sup> C Interface Status Register 2	8	08h
-	Not Used	9	09h
-	Not Used	10	0Ah
USP_H	User Stack Pointer (MSB)	11	0Bh
USP_L	User Stack Pointer (LSB)	12	0Ch
-	Not Used	13-20	0Dh-014h
PWM0_COUNT_IN_H	PWM/Timer 0 Counter Input Register (MSB)	21	015h
PWM0_COUNT_IN_L	PWM/Timer 0 Counter Input Register (LSB)	22	016h
PWM0_STATUS	PWM/Timer 0 Status Register	23	017h
PWM0_CAPTURE_H	PWM/Timer 0 Capture Register (MSB)	24	018h
PWM0_CAPTURE_L	PWM/Timer 0 Capture Register (LSB)	25	019h
PWM1_COUNT_IN_H	PWM/Timer 1 Counter Input Register (MSB)	26	01Ah
PWM1_COUNT_IN_L	PWM/Timer 1 Counter Input Register (LSB)	27	01Bh
PWM1_STATUS	PWM/Timer 1 Status Register	28	01Ch
PWM1_CAPTURE_H	PWM/Timer 1 Capture Register (MSB)	29	01Dh
PWM1_CAPTURE_L	PWM/Timer 1 Capture Register (LSB)	30	01Eh
-	Not Used	31-35	01Fh-023h
SCI_IN	Serial Communication Interface RX data Input Register	36	024h
SCI_STATUS	Serial Communication Interface Status Register	37	025h
FLAGS	Flag Register	38	026h
AD_OVF	10-bit A/D Converter Overflow Register	39	027h
IAP_SR	In Application Programming Status Register	40	028h

Table 3.1 Input Registers

Mnemonic	Description	Address	
CHAN0_H	10-bit A/D Converter Channel 0 data Input Register (MSB)	41	029h
CHAN0_L	10-bit A/D Converter Channel 0 data Input Register (LSB)	42	02Ah
CHAN1_H	10-bit A/D Converter Channel 1 data Input Register (MSB)	43	02Bh
CHAN1_L	10-bit A/D Converter Channel 1 data Input Register (LSB)	44	02Ch
CHAN2_H	10-bit A/D Converter Channel 2 data Input Register (MSB)	45	02Dh
CHAN2_L	10-bit A/D Converter Channel 2 data Input Register (LSB)	46	02Eh
CHAN3_H	10-bit A/D Converter Channel 3 data Input Register (MSB)	47	02Fh
CHAN3_L	10-bit A/D Converter Channel 3 data Input Register (LSB)	48	030h
CHAN4_H	10-bit A/D Converter Channel 4 data Input Register (MSB)	49	031h
CHAN4_L	10-bit A/D Converter Channel 4 data Input Register (LSB)	50	032h
CHAN5_H	10-bit A/D Converter Channel 5 data Input Register (MSB)	51	033h
CHAN5_L	10-bit A/D Converter Channel 5 data Input Register (LSB)	52	034h
CHAN6_H	10-bit A/D Converter Channel 6 data Input Register (MSB)	53	035h
CHAN6_L	10-bit A/D Converter Channel 6 data Input Register (LSB)	54	036h
CHAN7_H	10-bit A/D Converter Channel 7 data Input Register (MSB)	55	037h
CHAN7_L	10-bit A/D Converter Channel 7 data Input Register (LSB)	56	038h

Table 3.2 Output Registers

Mnemonic	Description	Address	
PORT_A_OUT	Port A data Output Register	0	00h
PORT_B_OUT	Port B data Output Register	1	01h
PORT_C_OUT	Port C data Output Register	2	02h
-	Not Used	3	03h
-	Not Used	4	04h
SPI_OUT	Serial Peripheral Interface data Output Register	5	05h
I2C_OUT	I <sup>2</sup> C Interface data Output Register	6	06h
PWM0_COUNT_OUT_H	PWM/Timer 0 Counter Output Register (MSB)	7	07h
PWM0_COUNT_OUT_L	PWM/Timer 0 Counter Output Register (LSB)	8	08h
PWM0_RELOAD_H	PWM/Timer 0 Reload Register (MSB)	9	09h
PWM0_RELOAD_L	PWM/Timer 0 Reload Register (LSB)	10	0Ah
PWM1_COUNT_OUT_H	PWM/Timer 1 Counter Output Register (MSB)	11	0Bh
PWM1_COUNT_OUT_L	PWM/Timer 1 Counter Output Register (LSB)	12	0Ch
PWM1_RELOAD_H	PWM/Timer 1 Reload Register (MSB)	13	0Dh
PWM1_RELOAD_L	PWM/Timer 1 Reload Register (LSB)	14	0Eh
SCI_OUT	Serial Communication Interface TX data Output Register	23	017h

Table 3.3 Option Bytes

Mnemonic	Description	Address	
OSC_CR	Oscillator Control Register	0	00h
CLK_SET	Clock Parameters	1	01h
OSC_SET	Oscillator Set-Up	2	02h
PLDV_CR	Programmable Low Voltage Detector Control Register	3	03h
WDT_EN	HW/SW Watchdog selector	4	04h
PG_LOCK	First Page Write Protected	5	05h
PG_UNLOCK	First Page not Write Protected	6	06h
WAKEUP	Wake Up from Halt Time	7	07h

Table 3.4 Configuration Registers

Mnemonic	Description	Address	
INT_MASK	Interrupt Mask Register	0	00h
INT_POL	Interrupts Polarity	1	01h
INT_PRL_H	Interrupt Priority Register (higher priority)	2	02h
INT_PRL_M	Interrupt Priority Register (medium priority)	3	03h
INT_PRL_L	Interrupt Priority Register (lower priority)	4	04h
USTP_H	User Stack Top Pointer (MSB)	5	05h
USTP_L	User Stack Top Pointer (LSB)	6	06h
WDT_CR	Watchdog Configuration Register	7	07h
AD_CR1	10-bit A/D Converter Control Register 1	8	08h
PWM0_CR1	PWM/Timer 0 Configuration Register 1	9	09h
PWM0_CR2	PWM/Timer 0 Configuration Register 2	10	0Ah
PWM0_CR3	PWM/Timer 0 Configuration Register 3	11	0Bh
PWM1_CR1	PWM/Timer 1 Configuration Register 1	12	0Ch
PWM1_CR2	PWM/Timer 1 Configuration Register 2	13	0Dh
-	Not Used	14	0Eh
-	Not Used	15	0Fh
I2C_CR	I <sup>2</sup> C Interface Control Register	16	010h
I2C_CCR	I <sup>2</sup> C Interface Clock Control Register	17	011h
I2C_OAR1	I <sup>2</sup> C Interface Own Address Register 1	18	012h
I2C_OAR2	I <sup>2</sup> C Interface Own Address Register 2	19	013h
SPI_CR	Serial Peripheral Interface Control Register	20	014h
SPI_STATUS_CR	Serial Peripheral Interface Control-Status Register	21	015h
SCI_CR1	Serial Communication Interface Control Register 1	22	016h
SCI_CR2	Serial Communication Interface Control Register 2	23	017h
PORT_A_PULLUP	Port A Pull Up enable/disable Register	24	018h
PORT_A_OR	Port A Option Register	25	019h

Table 3.4 Configuration Registers

Mnemonic	Description	Address	
PORT_A_DDR	Port A Data Direction Register	26	01Ah
PORT_A_AF	Port A Alternate Function selection Register	27	01Bh
PORT_B_PULLUP	Port B Pull Up enable/disable Register	28	01Ch
PORT_B_OR	Port B Option Register	29	01Dh
PORT_B_DDR	Port B Data Direction Register	30	01Eh
PORT_B_AF	Port B Alternate Function selection Register	31	01Fh
PORT_C_PULLUP	Port C Pull Up enable/disable Register	32	020h
PORT_C_OR	Port C Option Register	33	021h
PORT_C_DDR	Port C Data Direction Register	34	022h
PORT_C_AF	Port C Alternate Function selection Register	35	023h
-	Not Used	36-42	024h-02Ah
SCI_CR3	Serial Communication Interface Control Register 3	43	02Bh
SSP_H	System Stack Pointer (MSB)	44	02Ch
SSP_L	System Stack Pointer (LSB)	45	02Dh
CPU_CLK	CPU Clock Prescaler	46	02Eh
AD_CR2	10-bit A/D Converter Control Register 2	47	02Fh

## 4 MEMORY PROGRAMMING

ST52F510/F513/F514 provides an on-chip user programmable non-volatile memory, which allows fast and reliable storage of user data.

Program/Data Memory addressing space is composed by a Single Voltage Flash Memory and a RAM memory bench. The ST52F513/514 devices also have a Data EEPROM bench to store permanent data with long term retention and a high number of write/erase cycles.

All the Program Data memory addresses can execute code, including RAM and EEPROM benches.

The memory is programmed by setting the  $V_{pp}$  pin equal to  $V_{dd2}$ . Data and commands are transmitted through the I<sup>2</sup>C serial communication protocol. The same procedure is used to perform "In-Situ" the programming of the device after it is mounted in the user system. Data can also be written in run-time with the In-Application Programming (IAP).

The Memory can be locked by the user during the programming phase, in order to prevent external operation such as reading the program code and assuring protection of user intellectual property. Flash and EEPROM pages can be protected by unintentional writings.

**Remark:** the memory contents are protected by the Error Correction Code (ECC) algorithm that uses a 4-bit redundancy to correct one bit errors.

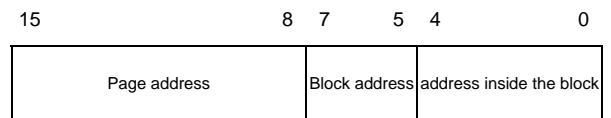
**Warning:** when entering the ISP, the default values for Option Bytes are considered, so a Voltage Supply higher than the PLVD lower threshold must be applied to program the device.

### 4.1 Program/Data Memory Organization

The Program/Data Memory is organized as described in Section 3.3. The various sales types have different amounts of each type of memory. Table 4.1 describes the memory benches amount and page allocation for each sales type.

The addressing spaces are organized in pages of 256 bytes. Each page is composed by blocks of 32 bytes. Memory programming is performed one block at a time in order to speed-up the programming time (about 2.5 ms per block).

The whole location address is composed as follows:



**Table 4.1 Sales Type Memory Organization**

Device	Flash Memory		RAM Memory		EEPROM Memory	
	Amount	Pages	Amount	Page	Amount	Page(s)
ST52F510c2p6	4096 bytes	0 to 15	256 bytes	32	-	-
ST52F510c3p6	8192 bytes	0 to 31	256 bytes	32	-	-
ST52F513c2p6	3840 bytes	0 to 14	256 bytes	32	256 bytes	15
ST52F513c3p6	7936 bytes	0 to 30	256 bytes	32	256 bytes	31
ST52F514c1p6	4096 bytes	0 to 15	256 bytes	32	1024 bytes	16-19
ST52F514c3p6	4096 bytes	0 to 15	256 bytes	32	4096 bytes	16-31

legend:

c: Y=16 pins, F=20 pins, G=28 pins, K=32/34 pin

p: B=DIP, M=SO, T=TQFP

**4.2 Memory Programming**

The Programming procedure writes the user program and data into the Flash Memory, EEPROM and Option Bytes. The programming procedures are entered by setting the  $V_{pp}$  pin equal to  $V_{dd}$  and releasing the Reset signal. The following pins are used in Programming mode:

- $V_{pp}$  used to switch to programming mode
- $V_{DD}$  device supply
- $V_{SS}$  device ground
- RESET device reset
- SCL I<sup>2</sup>C serial clock
- SDA I<sup>2</sup>C serial data

During the device programming, the internal clock is used, so the OSCin and OSCout pins don't have to be considered.

**4.2.1 Programming Mode start.** The following sequence starts the Programming Mode:

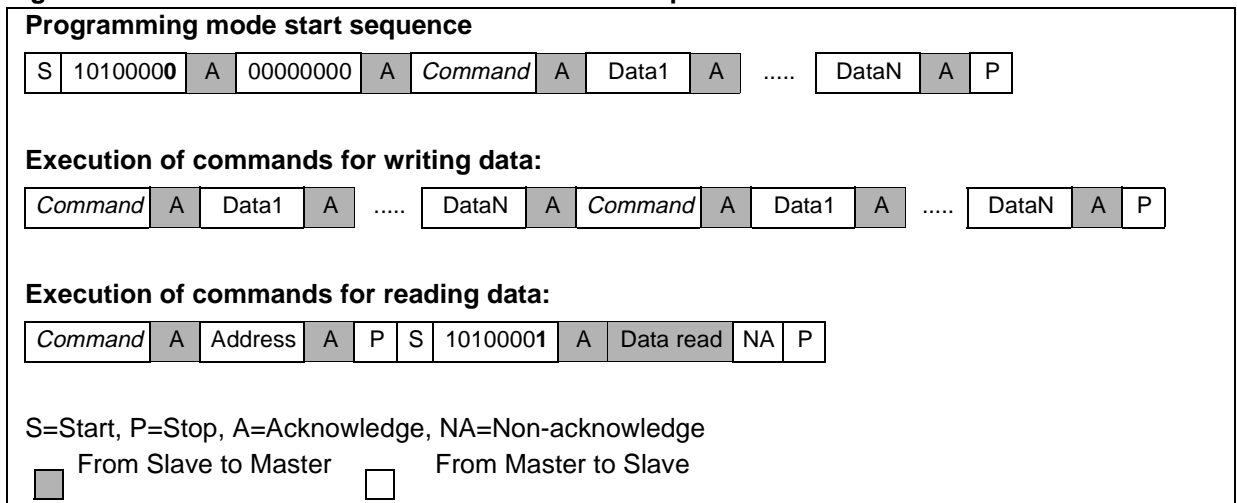
1.  $V_{pp}$  is set to  $V_{DD}$
2. The device is Reset (RESET= $V_{SS}$ )
3. The Reset is released (RESET= $V_{DD}$ )
4. The internal oscillator starts at 10 MHz
5. The memory is turned on
6. The I<sup>2</sup>C Interface and Ports are initialized
7. The I<sup>2</sup>C Interface is configured to work as Slave, Receiver, 7-bit address and waits for data
8. The Start signal is sent to the chip followed by the Slave Address 1010000 and the direction bit set to 0 (the addressed slave waits for data). The device sends the acknowledge
9. The Programming Mode code 00000000 is sent and acknowledged
10. A command code is sent to the device
11. The procedure related to the command is executed

**Table 4.2 Programming Mode Commands**

Command	Code	Data in	Data out	Erase	Description
BlockWrite	00000001	32	-	Yes	Write the currently addressed block with the 32 bytes following the command. The Block locations are erased before being written.
ByteWrite	00000010	2	-	Yes	Write the byte addressed by the next data sent in the currently addressed page.
BlockErase	00000011	1	-	Yes	Erase the block addressed by 3 MSB of the next data sent and inside the currently addressed page.
ByteErase	00000100	1	-	Yes	Erase the byte addressed by the next data sent and inside the currently addressed page.
ByteRead	00000101	1	1	-	Read the byte addressed by the next data sent and inside the current page. The read data is sent by the device after the re-send of the Slave Address with the R/W bit changed.
GlobalErase	00001001	-	-	Yes	All the memory is erased.
FastBlockWrite	00001011	32	-	No	Write the currently addressed block with the 32 bytes following the command. The Block locations aren't erased.
SetPage	00001100	1	-	-	The currently addressed page is set with the next data sent.
ReadData	00001101	-	1	-	Read the memory location currently addressed. The read data is sent by the device after the command is acknowledged. The current memory absolute address is post-incremented.
IncBlock	00001111	-	-	-	The current block address is incremented modulo 8 (address 0 follows after address 7 and the Page is post-incremented)
ReadStatus	00010011	-	1	-	This command is followed by a status data byte. Mostly used in error condition and to check if the device is locked



Figure 4.1 Commands and Data Communication Sequences



The generic procedure of commands execution, with the data communication in both directions is displayed in Figure 4.1.

**Remark:** the Slave Address 1010000 must be sent after a Stop (i.e. each time the data direction changes, to specify the R/W bit). For example: if a command to send data to the device has been executed, a command for receiving data must be followed by the slave address and the R/W bit must be set to 1. The Programming Mode code doesn't need to be specified again.

**Warning:** After entering the Programming Mode, the currently pointed address is the Page 48, Block 3, byte 0 (Lock Byte).

The list of the available commands in Programming Mode is showed in Table 4.2

**4.2.2 Fast Programming procedure.** The fastest way to program the device memory is the use of the *FastBlockWrite* command. The following procedure can be used to write the memory with a new program and new data, starting from the first memory location:

1. The Programming Mode is entered with the sequence described above
2. The memory is erased (all bits are put to 0) with the *GlobalErase* command. The device holds the SCL line low, releasing it after the command is completed (about 2 ms). This command also unlocks the device if locked.
3. The *FastBlockWrite* command is sent and the device acknowledges it
4. The 32 bytes of data to be written in the first memory Block are sent in a sequence. The device acknowledges each of them

5. After the device acknowledges the 32nd byte, it holds the SCL line until the parallel writing of the 32 byte is completed (about 2.5 ms)
6. The Block Pointer is incremented by sending the *IncBlock* command
7. The procedure is repeated from point 3 until there is data to be sent to the memory

**Note:** the Block Pointer assumes values between 0 to 7 (there are 8 blocks in a page). When the Block Pointer is equal to 7, the *IncBlock* command puts this pointer to 0 and increments the Page Pointer. The Page Pointer, after page writing is completed, doesn't have to be incremented in the procedure above described.

**4.2.3 Random data writing.** A single byte can be written in a specified memory location by using the following procedure:

1. The Programming Mode is entered with the sequence described in Section 4.2.1
2. The *SetPage* command is sent, followed by the page number where the data should be written
3. The *ByteWrite* command is sent followed by two bytes
4. The first bytes that follows the *ByteWrite* command is the address inside the pointed page where the data must be written.
5. The second byte is the data to be written
6. The device held the SCL line low until the data is not stored in the memory (about 4.5 ms: 2 ms for erasing and 2.5 for writing)

A similar procedure can be used to write a single block:

1. The *SetPage* command is sent, followed by the page number where the data should be written
2. The *IncBlock* command is sent as many times as the block number inside the page (for example: to address the block 3 the *IncBlock* must be sent 3 times)
3. The *WriteBlock* command is sent followed by the 32 data bytes to be written.
4. After the 32th byte is sent, the device holds the SCL line low until all the data are not stored in the memory (about 4.5 ms: 2 ms for erasing and 2.5 for writing; the same time for a single byte)

The procedures described previously can be repeated as many time as needed, without exiting from Programming Mode or re-sending the Slave Address again.

The commands *ByteErase* and *BlockErase*, used instead of *ByteWrite* and *BlockWrite*, erase (put all bit to 0) the specified memory location or block.

**4.2.4 Option Bytes Programming.** The Option Byte addresses cannot be accessed with a sequential procedure like the one described in Section 4.2.2. Actually, the pointers are automatically incremented up to the last block or address in page 31. A further increment sets all the pointers to 0.

The Option Byte addresses (located at page 48, block 0, addresses 0-7) must be accessed with a direct addressing procedure as the one described in Section 4.2.3.

If the Fast Programming procedure is used, it must be followed by a Random Block Writing procedure to program the Option Bytes. The other 24 bytes of the block can be written with dummy or user values. The blocks 0, 1, 2 and 3 of Page 48 can be used for writing data as well (see Section 4.5) and for locking the device (see Section 4.4).

Figure 4.2 Programming Procedures

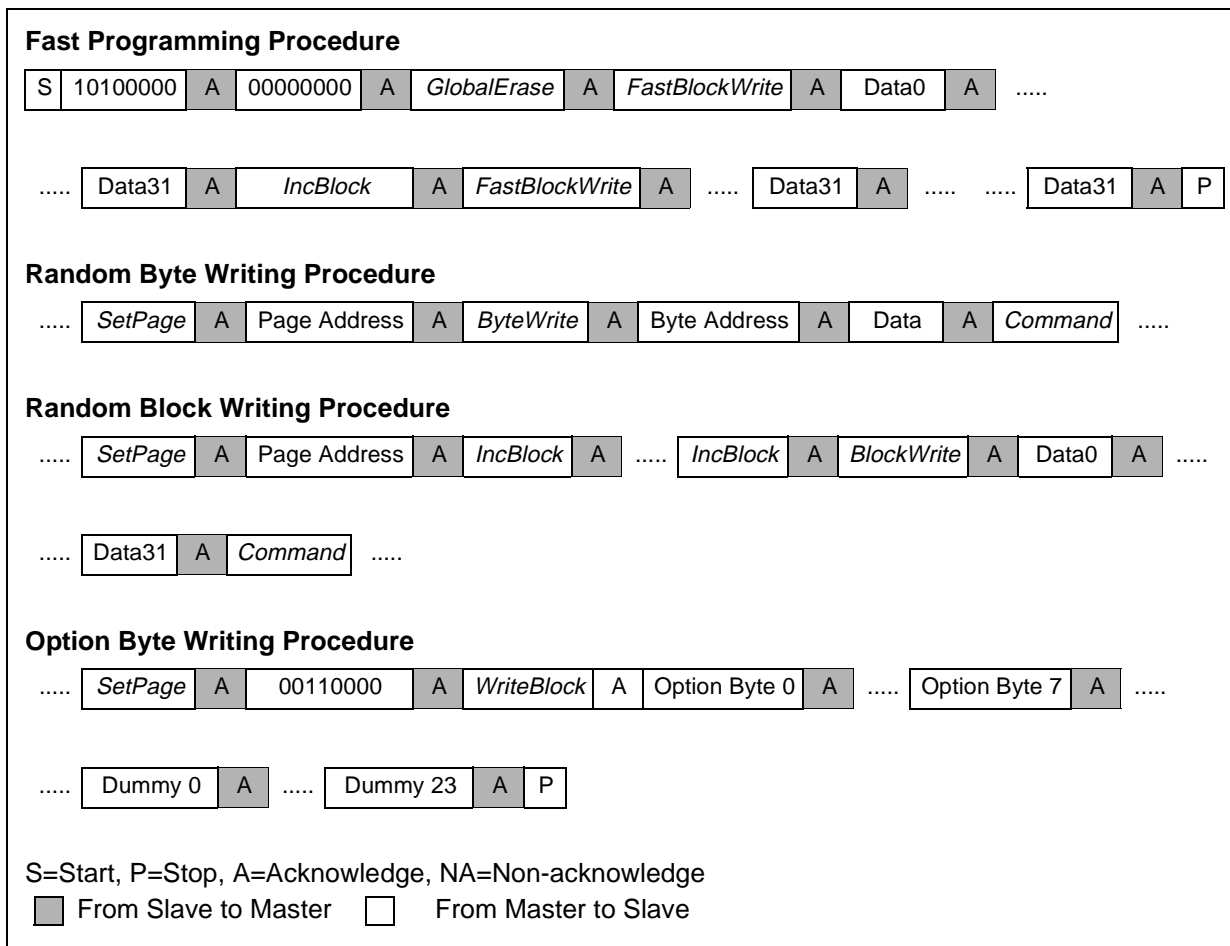
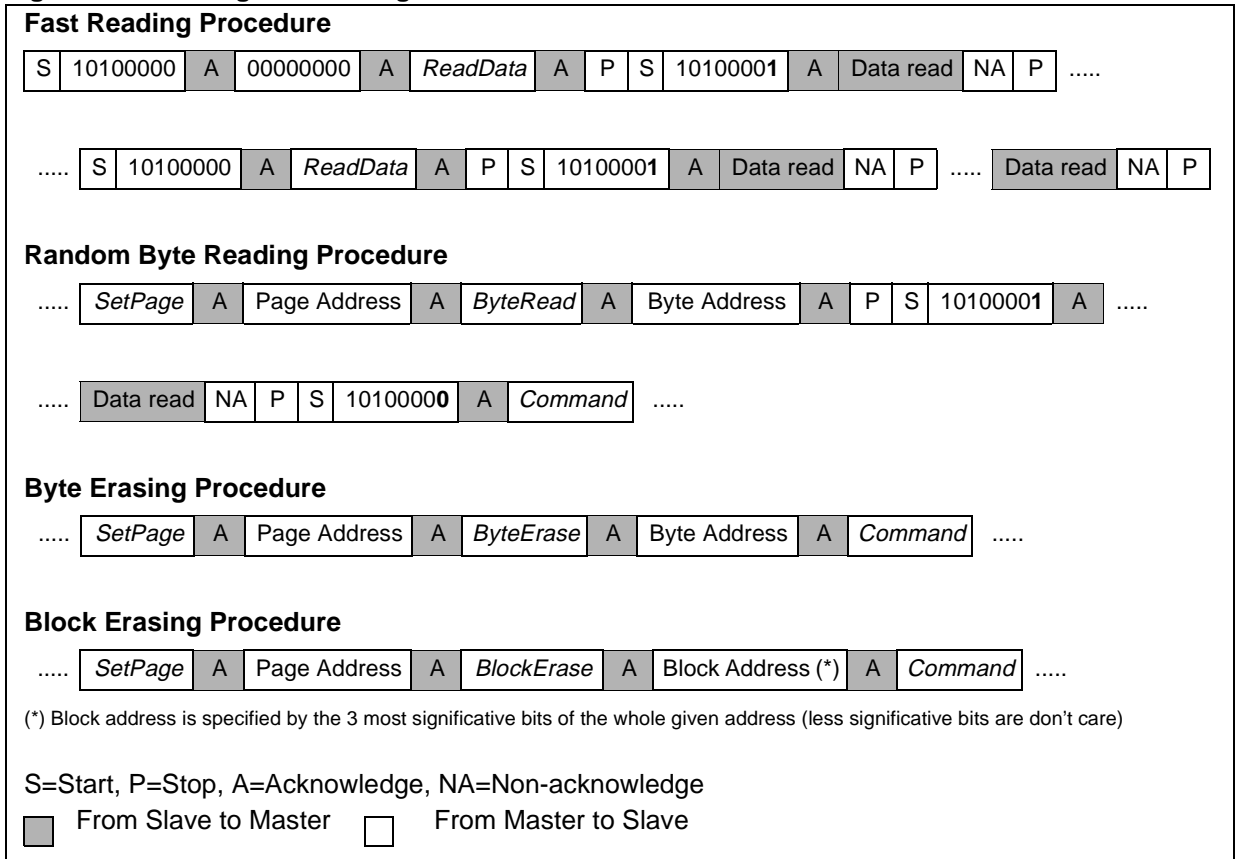


Figure 4.3 Reading and Erasing Procedures



### 4.3 Memory Verify

To verify the memory contents or just to read part of data stored in memory, the *ByteRead* and the *ReadData* command can be used. The first instruction needs the specification of the address; the second one allows the sequential reading of consecutive memory locations.

Since the device is "Slave" for the I<sup>2</sup>C protocol, after receiving a command for reading, it must be configured as Slave Transmitter to send the data. In order to do so, the Slave Address (1010000) must be sent again with the R/W byte set to 1, as stated by the communication protocol.

**4.3.1 Fast read procedure.** The memory can be read sequentially by using the following procedure:

1. The Programming mode is entered with the sequence described in Section 4.2.1
2. The pointers address the memory location 0
3. The *ReadData* command is sent and the device acknowledges it.
4. The Master generates a Stop condition followed by a Start condition

5. The Slave Address with the R/W byte set to 1 (10100001) is sent. The device receives the Slave Address and acknowledges it.
6. The device sends the data to be read in the serial data line SDA. The current absolute address is post-incremented.
7. The Master device doesn't send the acknowledge and generates a stop condition.
8. To read the next data, the Master generates a Start condition followed by the Slave Address with the R/W byte set to 0 (10100000). The device receives the Slave Address and acknowledges it.
9. The sequence restarts from point 3 until there is data to be read.

**Remark:** for the same reasons explained in Section 4.2.4 the Option Bytes cannot be read with this procedure: they can be read with a direct addressing procedure as the one explained in the next section.

**4.3.2 Random data reading.** To read a specified memory location, the following procedure should be used:

1. The Programming mode is entered with the sequence described in Section 4.2.1
2. The *SetPage* command is sent, followed to the page number where the data to be read is located
3. The *ByteRead* command is sent, followed by an address inside the page
4. The Master generates a Stop condition followed by a Start condition
5. The Slave Address with the R/W byte set to 1 (10100001) is sent. The device receives the Slave Address and acknowledges it.
6. The device sends the data to be read in the serial data line SDA.
7. The Master device doesn't send the acknowledge and generates a stop condition.
8. To send the next command, the Master should generate a Start condition followed by the Slave Address with the R/W byte set to 0 (10100000).

**4.4 Memory Lock**

The Program/Data Memory space can be locked to inhibit the reading of contents and protect the intellectual property.

To lock the device, the user must set all the bit of the Lock Byte to '1'. The Lock Byte is located on Page 48 (030h), Block 3, byte 0 inside the block i.e. byte 96 (060h) inside the page.

After writing 255 (0FFh) into the Lock Byte, with the procedure described in the Section 4.2.3, the memory is locked and the only command allowed are the following:

- *GlobalErase*: this command, writing '0' in all the memory, also unlock the device.
- *ReadData*: the only block that can be read is the Block 3 in Page 48 (030h); this allows the reading of the Lock Byte and the ID Code locations (see Section 4.5).
- *ReadStatus*: this command allows the detection of an error condition in Programming mode operation (see Section 4.6). It can also be used to check if the device is locked. The most significant bit return the Lock Bit (0=unlocked, 1=locked).

**Remark:** the Lock Byte is checked when entering the Programming Mode. For this reason after writing the Lock Byte, all the commands can be carried out until the Programming mode is exited.

**Figure 4.4 Device Lock Procedure**

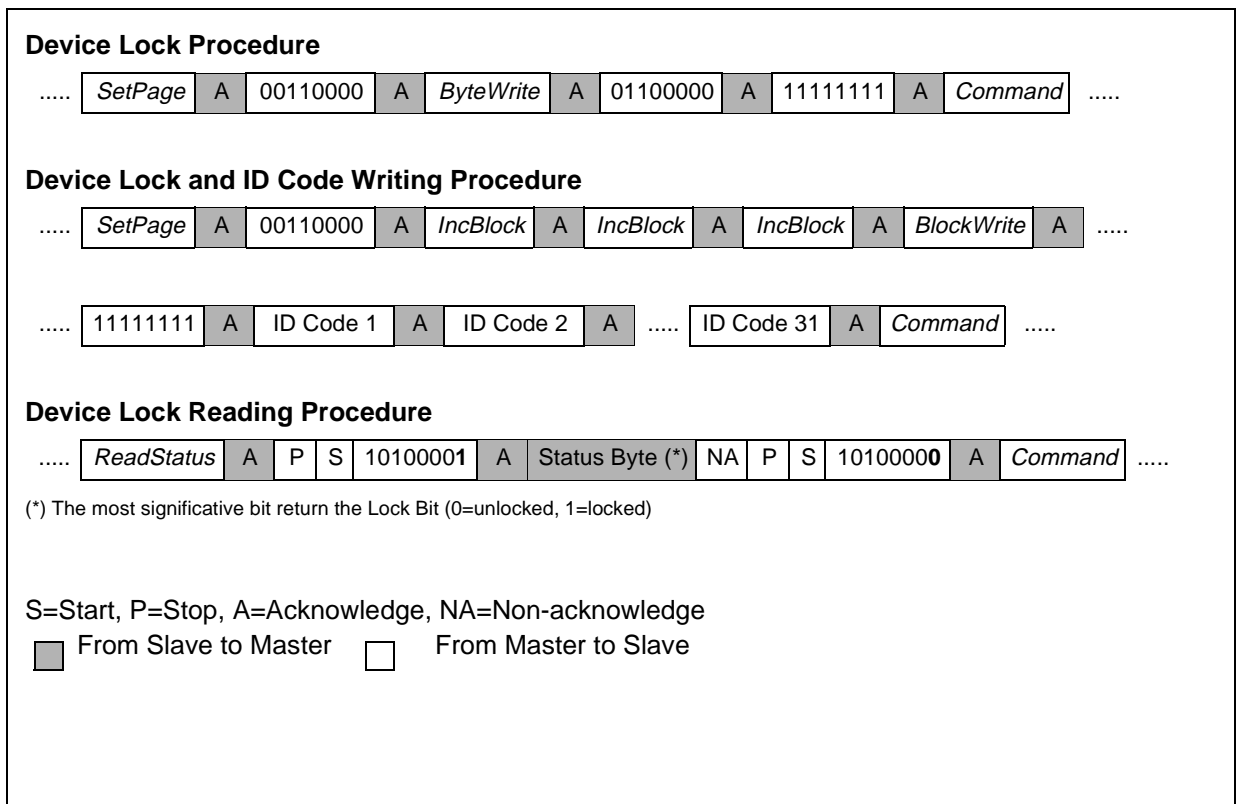
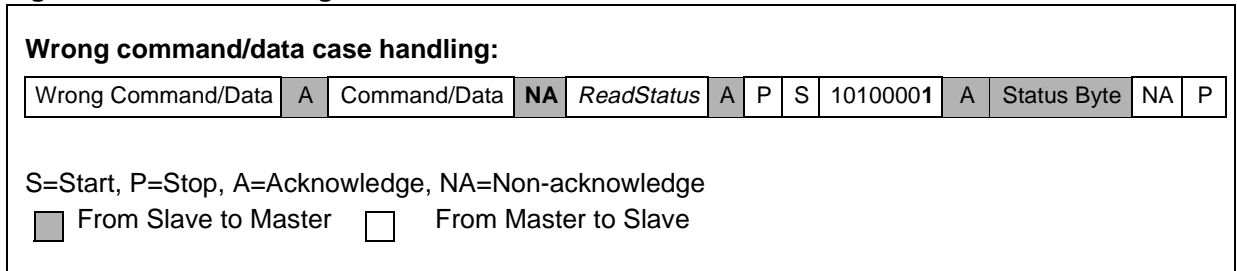


Figure 4.5 Error Handling Procedure



When the device is locked, if memory reading is attempted, with the exception of the Lock Byte and ID Code block, the device returns no data and an error sequence. If memory writing is attempted in any memory location, the device doesn't carry out the command and returns an error sequence.

To unlock the device the *GlobalErase* command must be executed before any writing or reading command.

#### 4.5 ID Code

Block 3 on Page 48 (030h) can also be read if the device is locked. The first byte of the block is the Lock Byte, the other 31 locations are available to the user for writing data, as for example identification codes to distinguish the firmware version loaded in the device.

The ID Code must be written before locking the device: after the device is locked it can only be read. The use of the Block writing procedure is the fastest way: both the ID Code and the Lock bytes are written together. The lock Bytes are sent first and then the 31 bytes of ID Code follow.

The blocks 0, 1 and 2 on Page 48 can be also be used for writing data, but they cannot be accessed when the device is locked.

**Note:** the ID Code cannot be modified if the device is locked: it can only be read.

**Table 4.3 Error codes**

Name	Code	Description
Device Locked	xyyyyyyy	x=lock bit (1=device locked), yyyyyyy=error code
Wrong Direction	x0000001	A transmit direction, not correct in the running sequence, has been set
Stop Missed	x0000010	The Master missed generating a necessary Stop Condition
Data Missing	x0000011	The Master missed to send necessary data to the device
Receive Error	x0000100	The data sent by the Master hasn't been received correctly by the device
Wrong Command	x0000101	The Master sent a wrong command code
Not Allowed	x0000110	A command not allowed when the device is locked has been sent
Wrong Mode	x0010000	A code different form the Programming mode code (00000000) has been sent

#### 4.6 Error cases

If a wrong command or data is sent to the device, it generates an error condition by not sending the acknowledge after the first successive data or command. Figure 4.5 shows the error sequence.

The error case can be handled by using the *ReadStatus* command. This command can be sent after the error condition is detected; the device returns a Status Byte containing the error code. The *ReadStatus* command sequence is showed in Figure 4.5. The list of the error codes is illustrated in Table 4.3.

**Remark:** after the *ReadStatus* command execution or after any error, the Start Sequence must be carried out before sending a new command.

The Most Significant Bit of the error codes indicates (when set to '1') that the memory is locked. When a command, that is not allowed when the memory is locked, is sent, the "Not Allowed" code is sent. If another code is sent with the MSB to '1' it indicates that the error condition is not caused by the memory lock, but by the event related with the code sent.

**Warning:** when the data writing into a non existing location is attempted, no error condition is generated. The user must take care in specifying the correct page address.

#### 4.7 In-Situ Programming (ISP)

The Program/Data Memory can be programmed using the ISP mode. This mode allows the device to be programmed when it is mounted in the user application board.

This feature can be implemented by adding a minimum number of components and board impact.

The programming procedures and pins used are identical to the ones described before for the standard Programming Mode. All the features previously described in this chapter are applicable in ISP mode.

If RESET, SCL and SDA pins are used in the user application board for other purposes, it is recommended to use a serial resistor to avoid a conflict when the other devices force the signal level.

The ISP can be applied by using the standard tools for the device programming. The ISP can be applied by using the standard tools for the device programming. The ST52F510 Starter Kit supplies a cable to perform the ISP. The user application board should supply a suited connector type for the cable (see Starter Kit User Manual).

#### 4.8 In-Application Programming (IAP)

The In Application Programming Mode (IAP) allows the writing of user data in the Flash and EEPROM memories when the user program is running.

There are two ways to write data in IAP mode: single byte write and Block write. Both procedures take about 4.5 ms to complete the writing: the Block write allows the writing of 32 byte in parallel.

**Remark:** during data writing, the execution of the user program is stopped until the procedure is completed. Interrupt requests stop the writing operation and the data may be not stored. The bit ABRT in the IAP\_SR Input register signals that the data writing hasn't been completed. To assure writing completion, the user should globally disable the interrupts (UDGI instruction) before starting IAP data writing.

**4.8.1 Single byte write.** Writing of a single byte in the Non-Volatile Program/Data memory is performed by using the LDER instruction (both direct and indirect addressing). The memory page should be indicated before the LDER instruction with the PGSET or PGSETR instruction. The byte address inside the page is specified by the LDER instruction itself.

As soon as the instruction is executed, the data writing starts and is performed in about 4.5 ms.

**4.8.2 Block write.** This procedure allows the writing of 32 bytes in parallel. These bytes should belong to the same block.

Before the writing in the Program/Data memory, data must be buffered in the Register File in the first 32 locations (0-31, 00h-020h) by using the normal instructions to load the Register File locations.

Then the data writing starts by using the BLKSET instruction. The destination block is addressed by specifying the memory page with the PGSET or PGSETR instruction before to start the writing; the block inside the page is addressed with the argument of the BLKSET instruction.

Example:

```
PGSET 5
BLKSET 4
```

This instruction sequence writes the contents of the first 32 bytes of the Register File in the locations 1408-1439 (0580h-059Fh).

**Warning:** the user should be careful in specifying the correct page and block: the addressing of an not existing block can cause the unwanted writing of a different block.

As soon as the BLKSET instruction is executed, the data writing starts and is performed in about 4.5 ms.

This procedure may also be used to write few data, taking in account that all the 32 byte are written in the block anyway.

#### 4.8.3 Memory Corruption Prevention.

The user can protect some pages (or all the memory) from unintentional writings. The only constraint is that the protected pages must be consecutive.

Two Option Bytes allow the specification of the page to be protected: PG\_LOCK (Option Byte 5) and PG\_UNLOCK (Option Byte 6). PG\_LOCK is used to specify the first protected page; PG\_UNLOCK is used to specify the first page not protected after the protected ones. The pages between the two addresses are protected.

When writing in a protected page is attempted, the procedure is aborted and the bit PRTCD of IAP\_SR Input register is set.

If the PG\_LOCK and PG\_UNLOCK have the same value, no page is protected. By default, the two Option Bytes are programmed with the value 0, so the memory is not write protected by default.

In Programming Mode the protection is not considered and the pages can be written unless the device is locked.

#### 4.8.4 Option Bytes.

##### First Protected Page (PG\_LOCK)

Option Byte 5 (05h)

Reset Value: 0000 0000 (00h)

7							0
LCK7	LCK6	LCK5	LCK4	LCK3	LCK2	LCK1	LCK0

Bit 7-0: **LCK7-0** First Page write protected

In this register the address of first page to be protected in writing is specified. The pages following this one are protected up to the page specified by the PG\_UNLOCK Option Byte (not included among the protected ones).

##### First Page not Protected (PG\_UNLOCK)

Option Byte 6 (06h)

Reset Value: 0000 0000 (00h)

7							0
UNLCK7	UNLCK6	UNLCK5	UNLCK4	UNLCK3	UNLCK2	UNLCK1	UNLCK0

Bit 7-0: **UNLCK7-0** First Page not write protected

In this register the address of first page not write protected after the protected ones is specified. The pages following this one aren't protected.

#### 4.8.5 Input Register.

##### IAP Status Register (IAP\_SR)

Input Register 40 (028h) Read only

Reset Value: 0000 0000 (00h)

7						0	
-	-	-	-	-	-	PRTCD	ABRT

Bit 7-2: Not Used

Bit 1: **PRTCD** Page Protected

0: The writing has been completed

1: The writing has been aborted because the page is protected.

Bit 0: **ABRT** Writing operation aborted

0: The writing has been completed

1: The writing has been aborted because an interrupt or another unspecified cause occurred.

The ABRT and PRTCD bits are reset after the next successful data writing in the Flash of EEPROM memory.

## 5 INTERRUPTS

The Control Unit (CU) responds to peripheral events and external events through its interrupt channels.

When such events occur, if the related interrupt is not masked and doesn't have a priority order, the current program execution can be suspended to allow the CU to execute a specific response routine.

Each interrupt is associated with an interrupt vector that contains the memory address of the related interrupt service routine. Each vector is located in the Program/Data Memory space at a fixed address (see Figure 3.2 Program/Data Memory Organization).

### 5.1 Interrupt Processing

If interrupts are pending at the end of an arithmetic or logic instruction, the interrupt with the highest priority is acknowledged. When the interrupt is acknowledged the flags and the current PC are saved in the stacks and the associated Interrupt routine is executed. The start address of this routine (Interrupt Vector) is located in three bytes of the Program/Data Memory between address 3 and 32 (03h-020h). See Table 5.1 for the list of the Interrupt Vector addresses.

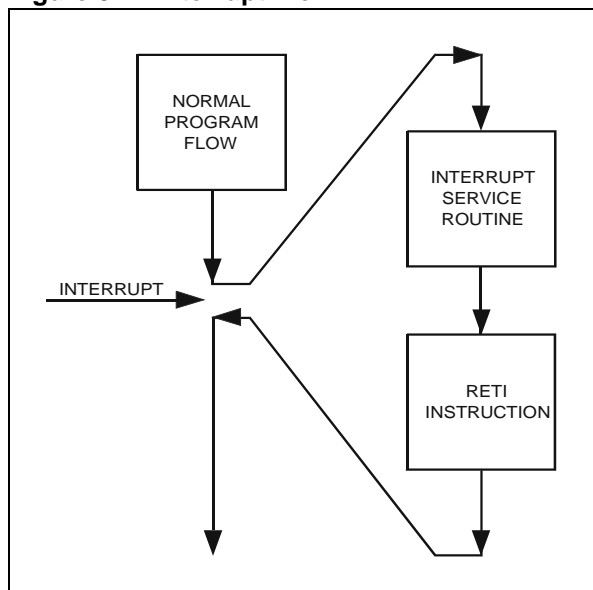
The Interrupt routine is performed as a normal code. At the end of each instruction, the CU checks if a higher priority interrupt has sent an interrupt request. An Interrupt request with a higher priority stops lower priority Interrupts. The Program Counter and the flags are stored in their own stacks.

With the instruction RETI (Return from Interrupt) the flags and the Program Counter (PC) are restored from the top of the stacks. These stacks have already been described in Paragraph 3.4.

An Interrupt request cannot stop fuzzy rule processing, but only after the end of a fuzzy rule or at the end of a logic or arithmetic instruction, unless a Global Interrupt Disable instruction has been executed before (see below).

**Remark:** A fuzzy routine can be interrupted only in the Main program. When a Fuzzy function is running inside another interrupt routine an interrupt request can cause side effects in the Control Unit. For this reason, in order to use a Fuzzy function inside an interrupt routine, the user **MUST** include the Fuzzy function between an UDGI (MDGI) instruction and an UEGI (MEGI) instruction (see the following paragraphs), in order to disable the interrupt request during the execution of the fuzzy function.

Figure 5.1 Interrupt Flow



### 5.2 Global Interrupt Request Enabling

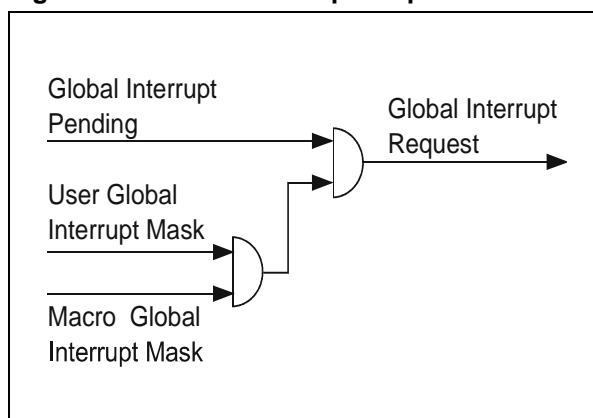
When an Interrupt occurs, it generates a Global Interrupt Pending (GIP). After a GIP a Global Interrupt Request (GIR) will be generated and Interrupt Service Routine associated with the interrupt with higher priority will start.

In order to avoid possible conflicts between the interrupt masking set in the main program, or inside high level language compiler macros, the GIP is put in AND through the User Global Interrupt Mask or the Macro Global Interrupt Mask (see Figure 5.2).

The UEGI/UDGI instruction switches the User Global Interrupt Mask enabling/disabling the GIR for the main program.

MEGI/MDGI instructions switch the Macro Global Interrupt Mask on/off in order to ensure that the macro will not be interrupted.

Figure 5.2 Global Interrupt Request





### 5.3 Interrupt Sources

ST52F510/F513/F514 manages interrupt signals generated by the internal peripherals or generated by software by the TRAP instruction or coming from the Port pins. There are two kinds of interrupts coming from the Port pins: the NMI and the Ports Interrupts.

NMI (Not Maskable Interrupt) is associated with pin PA7 when it is configured as Alternate Function. This interrupt source doesn't have a configurable level priority and cannot be masked. The fixed priority level is lower than the software TRAP and higher than all the other interrupts. The NMI can be configured to be active on the rising or the falling edge.

The Port Interrupts sources are connected with Port A and Port B pins. The pins belonging to the same Port are associated with the same interrupt vector: there is one vector for Port A and one for Port B. In order to use one port pin as interrupt, it must be configured as an interrupt source (see I/O Ports chapter). In this manner, up to 16 Port Interrupt sources are available. By reading the Port the sources that belong to the same Port can be discriminated. The Port Interrupts can be configured to be active on the rising or the falling edge.

**Warning:** changing the NMI or Port Interrupt polarity an interrupt request is generated.

All the interrupt sources are filtered, in order to avoid false interrupt requests caused by glitches.

The Trap instruction is something between a interrupt and a call: it generated an interrupt request at top priority level and the control is passed to the associated interrupt routine which vector is located in the fixed addresses 30-32. This routine cannot be interrupted and it is serviced even if the interrupts are globally disabled.

**Note:** Similarly to the CALL instruction, after a TRAP the flags are not stacked.

### 5.4 Interrupt Maskability and Priority Levels

Interrupts can be masked by the corresponding INT\_MASK Configuration Register 0 (00h). An interrupt is enabled when the mask bit is "1". Vice versa, when the bit is "0", the interrupt is masked and the eventual requests are kept pending.

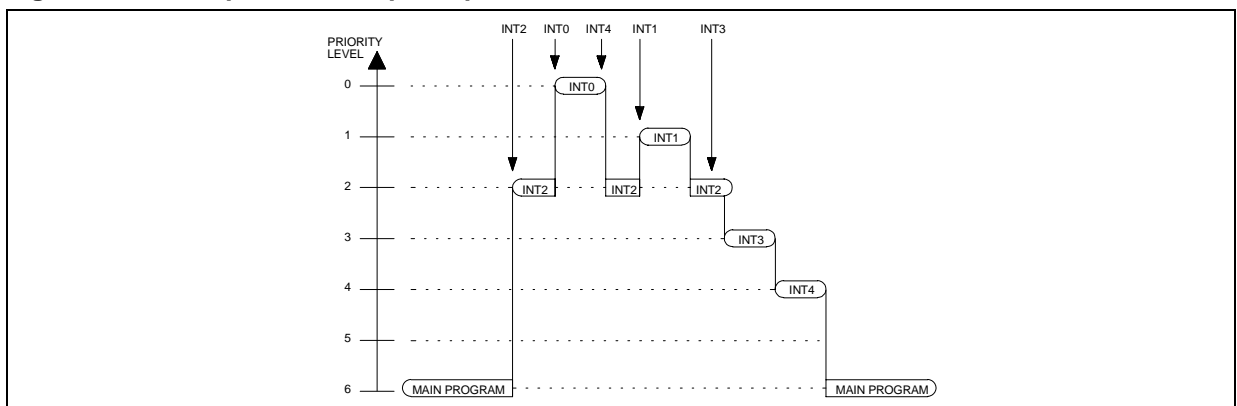
All the interrupts, with the exception of the NMI and TRAP that have fixed level priority, have a configurable priority level. The configuration of the priority levels is completed by writing three consecutive Configuration Registers: INT\_PRL\_H, INT\_PRL\_M, INT\_PRL\_L, addresses from 2 to 4 (02h-04h). The 24 bits of these registers are divided into 8 groups of three bits: each group is associated with a priority level. The three bits of each group are written with the code number associated with the interrupt source. See Table 5.1 to know the codes.

**Remark:** The priority levels Configuration Registers must be programmed with different values for each 3-bit groups to avoid erroneous operation. For this reason the Interrupt priority must be fixed at the beginning of the main program, because the reset values of the Configuration Registers correspond to an undefined configuration (all zeros). During program execution the interrupt priority can only be modified within the Main Program: it cannot be changed within an interrupt service routine.

### 5.5 Interrupt RESET

When an interrupt is masked, all requests are not acknowledged and remain pending. When the pending interrupt is enabled it is immediately serviced. This event may be undesired; in order to avoid this a RINT instruction may be inserted followed by the code number that identifies the interrupt to reset the pending request. See Table 5.1 to know the codes.

Figure 5.3 Example of Interrupt Requests



5.6 Register Description

**Interrupt Mask Register (INT\_MASK)**

Configuration Register 0 (00h) Read/Write

Reset Value: 0000 0000 (00h)

7							0
MSKPB	MSKPA	MSKI2C	MSKSPI	MSKSCI	MSKT1	MSKT0	MSKAD

Bit 7: **MSKPB** Interrupt Mask Port B

- 0: Port B interrupt masked
- 1: Port B interrupt enabled

Bit 6: **MSKPA** Interrupt Mask Port A

- 0: Port A interrupt masked
- 1: Port A interrupt enabled

Bit 5: **MSKI2C** Interrupt Mask I<sup>2</sup>C Interface

- 0: I<sup>2</sup>C Interface interrupt masked
- 1: I<sup>2</sup>C Interface interrupt enabled

Bit 4: **MSKSPI** Interrupt Mask SPI

- 0: SPI interrupt masked
- 1: SPI interrupt enabled

Bit 3: **MSKSCI** Interrupt Mask SCI

- 0: SCI interrupt masked
- 1: SCI interrupt enabled

Bit 2: **MSKT1** Interrupt Mask PWM/Timer 1

- 0: Pwm/Timer 1 interrupt masked
- 1: Pwm/Timer 1 interrupt enabled

Bit 1: **MSKT0** Interrupt Mask Pwm/Timer 0

- 0: Pwm/Timer 0 interrupt masked
- 1: Pwm/Timer 0 interrupt enabled

Bit 0: **MSKAD** Interrupt Mask A/D Converter

- 0: A/D interrupt masked
- 1: A/D interrupt enabled

**Interrupt Polarity Register (INT\_POL)**

Configuration Register 1 (01h) Read/Write

Reset Value: 0000 0000 (00h)

7							0
-	-	-	RESPOL	STRPOL	POLPB	POLPA	POLNMI

Bit 7-5: Not Used

Bit 4-3: See Timer 0 Registers Description

Bit 2: **POLPB** Port B Interrupt Polarity

- 0: The Port B interrupt is triggered on the rising edge of the applied external signal.
- 1: The Port B interrupt is triggered on the falling edge of the applied external signal.

Bit 1: **POLPA** Port A Interrupt Polarity

- 0: The Port A interrupt is triggered on the rising edge of the applied external signal.
- 1: The Port A interrupt is triggered on the falling edge of the applied external signal.

Bit 0: **POLNMI** Non Maskable Interrupt Polarity

- 0: The NMI is triggered on the rising edge of the applied external signal.
- 1: The NMI is triggered on the falling edge of the applied external signal.

**High Priority Register (INT\_PRL\_H)**

Configuration Register 2 (02h) Read/Write

Reset Value: 1111 1010 (0FAh)

7							0
PRL23	PRL22	PRL21	PRL20	PRL19	PRL18	PRL17	PRL16

**Medium Priority Register (INT\_PRL\_M)**

Configuration Register 3 (03h) Read/Write

Reset Value: 1100 0110 (0C6h)

7							0
PRL15	PRL14	PRL13	PRL12	PRL11	PRL10	PRL9	PRL8

**Low Priority Register (INT\_PRL\_L)**

Configuration Register 4 (04h) Read/Write

Reset Value: 1000 1000 (088h)

7							0
PRL7	PRL6	PRL5	PRL4	PRL3	PRL2	PRL1	PRL0

These three register are used to configure the priority level of each interrupt source. The 24 bits of these registers (PRL24-PRL0) are divided into 8 groups of three bits: each group is associated with a priority level (from level 1, the highest, to level 8, the lowest: level 0 is fixed for the NMI that can be interrupted only by the TRAP) . The three bits of each group are written with the code number associated with the interrupt source (see Table 5.1).

**PRL2-PRL1:** Interrupt priority level 1 (highest)**PRL5-PRL3:** Interrupt priority level 2**PRL8-PRL6:** Interrupt priority level 3**PRL11-PRL9:**Interrupt priority level 4**PRL14-PRL12:** Interrupt priority level 5**PRL17-PRL15:** Interrupt priority level 6**PRL20-PRL18:** Interrupt priority level 7**PRL23-PRL21:** Interrupt priority level 8 (lowest)

Example: writing the code 110 into PRL8-PRL6 bits the priority level 3 is assigned to the Port A Interrupt.

**Warning:** the Priority Level configuration registers must be always configured.

**Table 5.1 Interrupt sources paramethers**

Interrupt Source	Priority type	PRL code	RINT code	Maskable	Vector Addresses
A/D Converter	Programmable	000	0	Yes	3-5 (03h-05h)
PWM/Timer 0	Programmable	001	1	Yes	6-8 (06h-08h)
PWM/Timer 1	Programmable	010	2	Yes	9-11 (09h-0Bh)
SCI	Programmable	011	3	Yes	12-14 (0Ch-0Eh)
SPI	Programmable	100	4	Yes	15-17 (0Fh-011h)
I <sup>2</sup> C Interface	Programmable	101	5	Yes	18-20 (012h-014h)
Port A	Programmable	110	6	Yes	21-23 (015h-017h)
Port B	Programmable	111	7	Yes	24-26 (018h-01Ah)
NMI	Fixed	-	8	No	27-29 (01Bh-01Dh)
TRAP	Fixed to highest	-	-	No	30-32 (01Eh-020h)

6 CLOCK, RESET & POWER SAVING MODES

6.1 Clock

The ST52F510/F513/F514 Clock Generator module generates the internal clock for the internal Control Unit, ALU and on-chip peripherals. The Clock is designed to require a minimum of external components.

ST52F510/F513/F514 devices supply the internal oscillator in four clock modes:

- External oscillator
- External clock
- External RC oscillator
- Internal clock

The device always starts in internal clock mode, excluding any external clock source. After the start-up phase the clock is configured according to the user definition programmed in the Option Byte 0 (OSC\_CR). The internal clock generator can supply an internal clock signal with a fixed frequency of 10 MHz ± 1%, without the need for external components. In order to obtain the maximum accuracy, the frequency can be calibrated by configuring the related Option byte 2 (OSC\_SET).

The external oscillator mode uses a quartz crystal or a ceramic resonator connected to OSCin and OSCout as illustrated in Figure 6.1. This figure also illustrates the connection of an external clock.

The ST52F510/F513/F514 oscillator circuit generates an internal clock signal with the same period and phase as the OSCIN input pin. The maximum frequency allowed is 24 MHz.

When the external oscillator is used, the loop gain can be adapted to the various frequencies values by configuring the three bits of the Option Byte 1 CLK\_SET (see Register Description, Table 6.2).

When an external clock is used, it must be connected to the pin OSCIN while OSCOUT can

be floating. In this case, Option Byte 1 bits must be written with 0 (000).

The crystal oscillator start-up time is a function of many variables: crystal parameters (especially  $R_s$ ), oscillator load capacitance (CL), IC parameters, environment temperature and supply voltage.

The crystal or ceramic leads and circuit connections must be as short as possible. Typical values for CL1, CL2 are 10pF for a 20 MHz crystal.

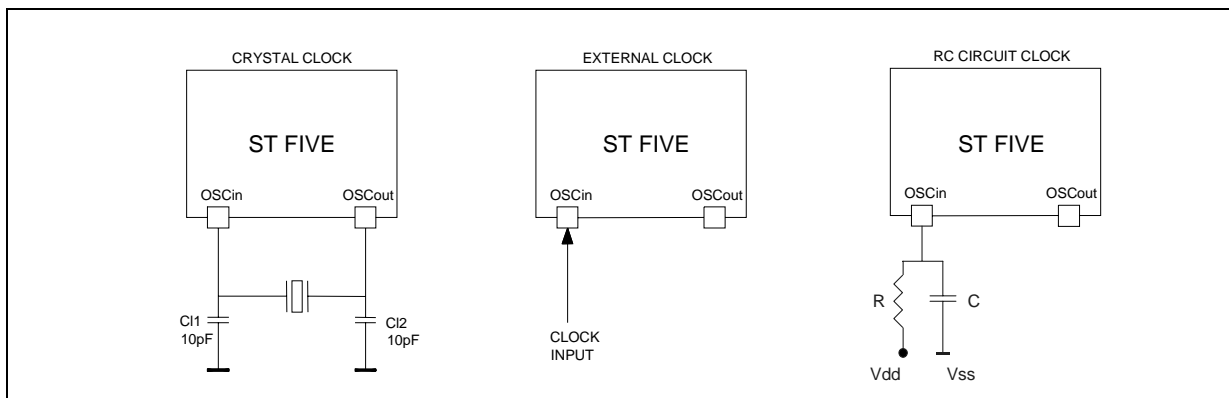
The clock signal can also be generated by an external RC circuit offering additional cost savings. Figure 6.1 illustrates the possible connections. Frequency is a function of resistor, capacitance, supply voltage and operating temperature; some indicative values when  $V_{dd}=5V$  and  $T=25^\circ$ , are shown in Table 6.1.

The clock signal generates two internal clock signals: one for the CPU and one for the peripherals. The CPU clock frequency can be reduced, in order to decrease current consumption, by setting the CPU\_CLK Configuration Register 46 (02Eh). The CPU clock can be reduced up to 64 times (see Register Description).

Table 6.1 RC Oscillator indicative frequencies

C (pF)	R(Ω)	f <sub>osc</sub> (KHz)	Variation
20 pF	9.5K	5000	6.6%
	10K	4870	7.1%
	20K	3000	5.3%
	50K	1360	3.3%
	100K	724	2.8%
100 pF	10K	1720	7.5%
	20K	926	8%
	50K	424	11.2%
	100K	248	15%

Figure 6.1 Oscillator Connections



## 6.2 Reset

Four Reset sources are available:

- RESET pin (external source)
- WATCHDOG (internal source)
- POWER ON Reset (Internal source)
- PLVD Reset (Internal source)

When a Reset event occurs, the user program restarts from the beginning.

**6.2.1 External Reset.** Reset is an input pin. An internal reset does not affect this pin. A Reset signal originated by external sources is recognized immediately. The RESET pin may be used to ensure Vdd has risen to a point where the ICU can operate correctly before the user program is run. Reset must be set to Vdd in working mode.

A Pull up resistor of 100 KΩ guarantees that the RESET pin is at level “1” when no HALT or Power-On events occur. If an external resistor is connected to the RESET pin a minimum value of 10KΩ must be used.

**6.2.2 Reset Procedures.** After the Reset pin is set to Vdd or following a Power-On Reset event, the device is not started until the internal supply voltage has reached the nominal level of 2.5 V (corresponding roughly to Vdd=2.8 V).

After this level has been reached, the internal oscillator (10 MHz) is started and a delay period of 4.096 clock cycles is initiated, in order to allow the oscillator to stabilize and to ensure that recovery has taken place from the Reset state.

If the device has been configured to work with the internal clock, the user program starts, otherwise the Option Byte 7 (WAKEUP) is read and another count starts before running the user program. The duration of the count depends on the contents of the Option Byte 7 (WAKEUP), that works as a prescaler, according to the following formula:

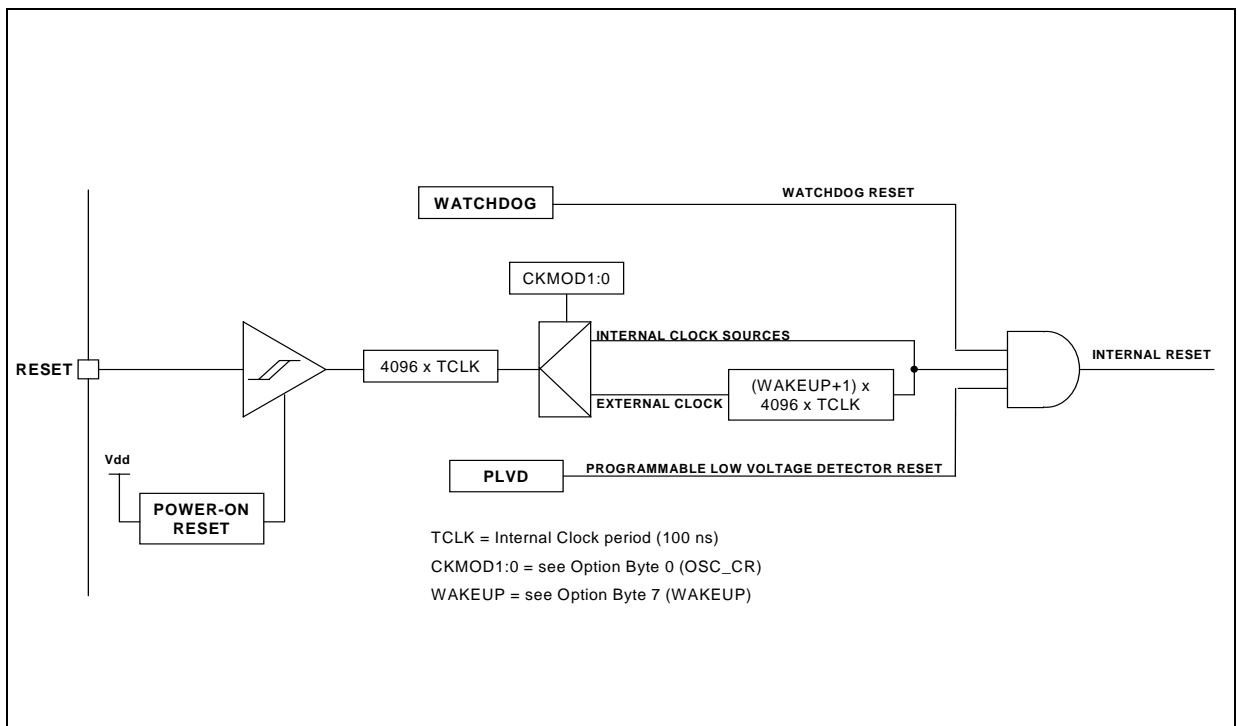
$$Delay = 4096 \times (WAKEUP + 1) \times T_{clk}$$

This delay has been introduced in order to ensure that the oscillator has become stable after its restart.

If the Reset is generated by the PLVD or the Watchdog, the oscillator is not turned off; for this reason the CPU is then restarted immediately, without the delay.

After a RESET procedure is completed, the core reads the instruction stored in the first 3 bytes of the Program/Data Memory, which contains a JUMP instruction to the first instruction of the user program. The Assembler tool automatically generates this Jump instruction with the first instruction address.

Figure 6.2 Reset Block Diagram



### 6.3 Programmable Low Voltage Detector

The on-chip Programmable Low Voltage Detector (PLVD) circuit prevents the processor from falling into an unpredictable status if the power supply drops below a certain level.

When Vdd drops below the detection level, the PLVD causes an internal processor Reset that remains active as long as Vdd remains below the trigger level.

The PLVD resets the entire device except the Power-on Detector and the PLVD itself.

The PLVD can be enabled/disabled at reset by setting the Option Byte 3 (PLVD\_CR) bits.

When Vdd increases above the Trigger Level, the PLVD reset is deactivated and the user program is started from the beginning.

The detection levels are programmable by means of the Option Byte 3 (PLVD\_CR). There are three levels for the PLVD falling voltages (2.9V, 3.4V, 3.9V) and for rising voltages (3.1V, 3.65V, 4.2V). The hysteresis for each level are respectively 200 mV, 250 mV and 300 mV.

The PLVD circuit will only detect a drop if Vdd voltage stays below the safe threshold for at least 5µs before activation/deactivation of the PLVD in order to filter voltage spikes.

**Remark:** the PLVD function isn't active when it is in HALT mode. In that case the device is reset if the Vdd voltage stays below the threshold of 2 V.

### 6.4 Power Saving modes

There are two types of Power Saving modes: WAIT and HALT mode. These conditions may be entered by using the WAIT or HALT instructions.

**6.4.1 Wait Mode.** Wait mode places the ICU in a low power consumption status by stopping the CPU. All peripherals and the watchdog remain active. During WAIT mode the Interrupts are enabled. The ICU remains in Wait mode until an Interrupt or a RESET occurs, whereupon the Program Counter jumps to the interrupt service routine or, if a Reset occurs, to the beginning of the user program.

**6.4.2 Halt Mode.** Halt mode is the lowest ICU power consumption mode, which is entered by executing the HALT instruction. The internal oscillator is turned off, causing all internal processing to be terminated, including the operations of the on-chip peripherals. Halt mode cannot be used when the watchdog is enabled. If the HALT instruction is executed while the watchdog system is enabled, it will be skipped without modifying the normal CPU operations.

The ICU can exit Halt mode upon reception of an NMI, a Port Interrupt or a Reset. The internal oscillator (10 MHz) is started and a delay period of 4.096 clock cycles is initiated, in order to allow the oscillator to stabilize and to ensure that recovery has taken place from the Reset state.

If the device has been configured to work with the internal clock, the user program is started, otherwise the Option Byte 7 (WAKEUP) is read and another count is started before running the user program. The count duration depends on the contents of the Option Byte 7 (WAKEUP), that works as prescaler, according to the following formula:

$$Delay = 4096 \times (WAKEUP + 1) \times T_{clk}$$

This delay has been introduced in order to ensure that the oscillator has become stable after it is restarted.

After the start up delay, by exiting with the NMI or a Port interrupt, the CPU restarts operations by serving the associated interrupt routine.

**Note:** if the Port Interrupt is masked, the ICU doesn't exit the Halt mode with this interrupt.

Figure 6.3 WAIT Flow Chart

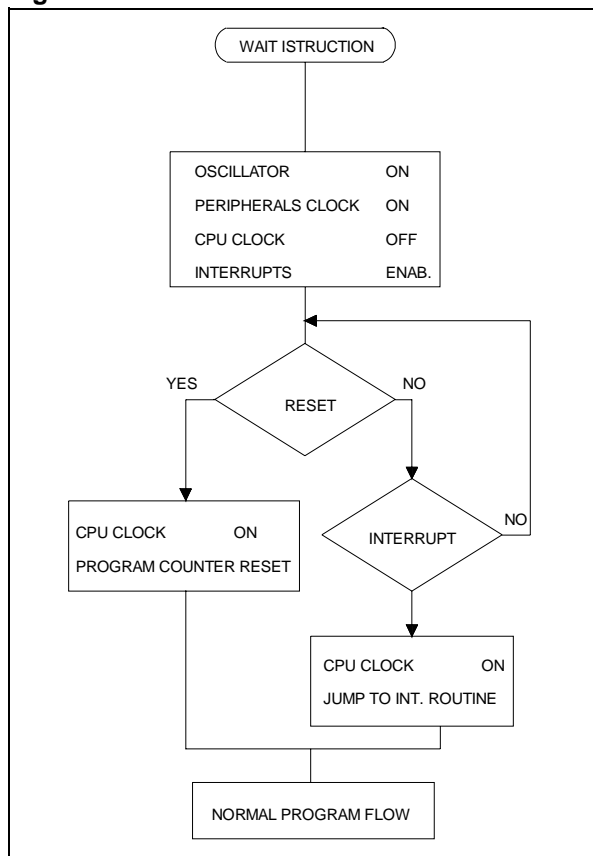
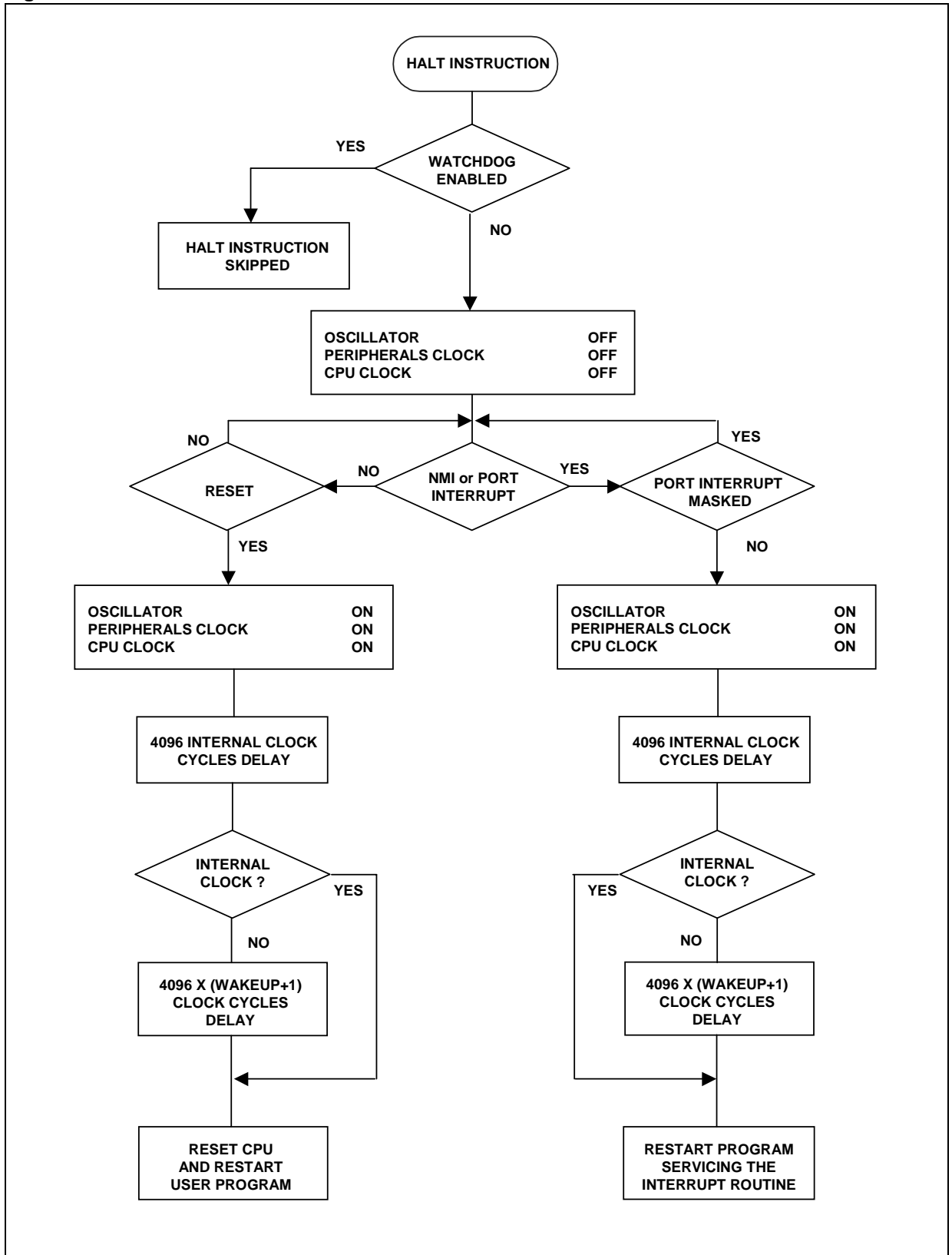


Figure 6.4 HALT Flow Chart



**6.5 Register Description**

The following section describes the Register which are used to configure the Clock, Reset and PLVD.

**6.5.1 Configuration Register.**

**CPU Clock Prescaler (CPU\_CLK)**

Configuration Register 46 (02Eh) Read/Write

Reset Value: 0000 0000 (00h)

7	0						
-	-	CPUCK5	CPUCK4	CPUCK3	CPUCK2	CPUCK1	CPUCK0

Bit 7-6: Not Used

Bit 5-0: **CPUCK5-0** CPU Clock Prescaler bits  
 The CPU Clock frequency is divided by a factor described in the following table

CPUCK5-0	CPU Clock
000000	$f_{CPU}=f_{OSC}$
000001	$f_{CPU}=f_{OSC}/2$
000010	$f_{CPU}=f_{OSC}/4$
000100	$f_{CPU}=f_{OSC}/8$
001000	$f_{CPU}=f_{OSC}/16$
010000	$f_{CPU}=f_{OSC}/32$
100000	$f_{CPU}=f_{OSC}/64$
others	$f_{CPU}=f_{OSC}/64$

**6.5.2 Option Bytes.**

**Clock Mode (OSC\_CR)**

Option Byte 0 (00h)

Reset Value: 0000 0000 (00h)

7	0							
-	-	-	-	-	-	-	CKMOD1	CKMOD0

Bit 7-2: Not Used

Bit 1-0: **CKMOD1-0** Clock Mode  
 00: Internal Oscillator  
 01: External Clock or quartz  
 1x: External RC oscillator

**External Clock Parameters (CLK\_SET)**

Option Byte 1 (01h)

Reset Value: 0000 0000 (00h)

7	0							
-	-	-	-	-	-	CKPAR2	CKPAR1	CKPAR0

Bit 7-3: Not Used

Bit 2-0: **CKPAR2-0** Oscillator Gains

These three bits enable/disable the loop gains when a external clock or quartz are used for generating the clock. The following table describes the possible configuration options. Table 6.2 illustrates the recommended values for the most common frequencies used, time to start the oscillations and the settling time to have a duty cycle of 40%-60% (at steady state it is 50%).

CKPAR2-0	Enabled Gain Stages
000	No Gains (External Clock Mode)
001	1 gain stage enabled
010	not allowed
011	3 gain stage enabled
100	not allowed
101	6 gain stage enabled
110	not allowed
111	8 gain stage enabled

**Warning:** If an External Clock is used instead of a quartz or ceramic resonator, it is recommended that no gain be enabled (CKPAR2-0=000) in order to lower the current consumption.





Table 6.2 Recommended Gains for the most common frequencies

Frequency	Recommend Gain Stages	CKPAR2-0	Oscillation Start Times	Settling Times for 40% duty-cycle
External Clock	0	000	-	-
1 MHz	1	001	367 $\mu$ s	27 $\mu$ s
4 MHz	1	001	84 $\mu$ s	10 $\mu$ s
8 MHz	3	011	75 $\mu$ s	9 $\mu$ s
10 MHz	3	011	79 $\mu$ s	5 $\mu$ s
12 MHz	6	101	110 $\mu$ s	8 $\mu$ s
16 MHz	6	101	352 $\mu$ s	7 $\mu$ s
20 MHz	8	111	165 $\mu$ s	11 $\mu$ s

(1) The recommended values have been chosen to have the best tradeoff between start time and current consumption. Higher gains give shorter Start times; lower gains give less current consumption.

(2) Indicative values by design at 25° Celsius,  $V_{DD}=2.6$  V. Not Tested in production.

### Internal Oscillator Calibration (OSC\_SET)

Option Byte 2 (02h)

Reset Value: 0001 0100 (14h)

7							0
-	-	OSPAR5	OSPAR4	OSPAR3	OSPAR2	OSPAR1	OSPAR0

Bit 7-6: Not Used

Bit 5-0: **OSPAR5-0** Internal Oscillator Parameters

These bits are used in order to calibrate the precision of the internal oscillator working at 10 MHz. The six bits enable some current generators with steps of 0.05  $\mu$ A corresponding to interval of frequency of 100KHz.

**Warning:** the maximum configuration value allowed is 101000 (40). The value corresponding to the 10 MHz by design is 010100 (20).

### PLVD Control Register (PLVD\_CR)

Option Byte 3 (03h)

Reset Value: 0000 0010 (02h)

7							0
-	-	-	-	-	-	PLVD1	PLVD0

Bit 7-2: Not Used

Bit 1-0: **PLVD1-0** PLVD detection levels

00: PLVD disabled

01: Medium detection level

10: Lowest detection level

11: Highest detection level

### Wake-Up Time Prescaler (WAKEUP)

Option Byte 7 (07h)

Reset Value: 0000 0000 (00h)

7							0
WK7	WK6	WK5	WK4	WK3	WK2	WK1	WK0

Bit 7-0: **WK7-0** Wake-up prescaler

This byte determines the time delay for the stabilization of the oscillator after an External Reset or a POR and after the wake-up from Halt. The time delay is computed according to the following formula:

$$Delay = 4096 \times (WAKEUP + 1) \times T_{clk}$$

**Warning:** If the internal clock is used as clock source the prescaler is not used.

7 I/O PORTS

7.1 Introduction

ST52F510/F513/F514 are characterized by flexible individually programmable multi-functional I/O lines. The ST52F510/F513/F514 supplies devices with up to 3 Ports (named from A to C) with up to 22 I/O lines.

Each pin can be used as a digital I/O or can be connected with a peripheral (Alternate Function). The I/O lines belonging to Port A and Port B can also be used to generate Port Interrupts.

The I/O Port pins can be configured in the following modes:

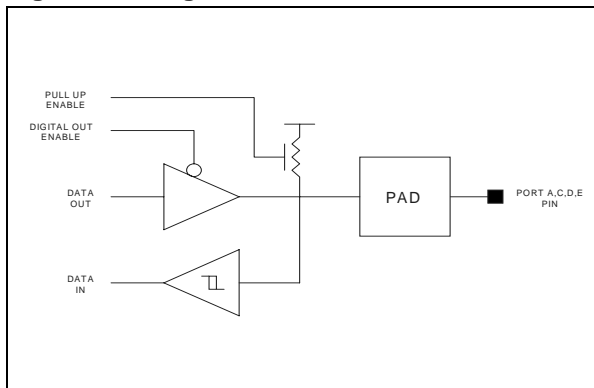
- Input high impedance (reset state)
- Input with pull-up
- Output with pull-up
- Output push-pull
- Output with weak pull-up
- Output open drain
- Interrupt with pull-up
- Interrupt without pull-up

These eight modes can be selected by programming three Configuration Registers for each Port. All the pins that belong to the same Port can be configured separately by setting the corresponding bits in the three registers (see Register Description).

To avoid side effects, the Configuration Registers register are latched only when the Direction Register (PORT\_x\_DDR) is written. For this reason this register must be always written when modifying the pin configuration.

All the I/O digital pins are TTL compatible and have a Schmitt Trigger. The output buffer can supply high current sink (up to 8mA).

Figure 7.1 Digital Pin



7.2 Input Mode

The pins configured as input can be read by accessing the corresponding Port Input Register by means of the LDRI instruction. The addresses for Port A , B and C are respectively 0 (00h), 1 (01h), and 2 (02h).

When executing the LDRI instruction all the signals connected to the input pins of the Port are read and the logical value is copied in the specified Register File location. If some pins are configured in output, the port buffer contents, which are the last written logical values in the output pins, are read.

7.3 Output Mode

The pins configured as output can be written by accessing the corresponding Port Output Register by means of the LDPR, LDPI and LDPE instructions. The addresses for Port A , B and C are respectively, 0 (00h), 1 (01h), and 2 (02h).

When executing the above mentioned instructions, the Port buffer is written and the Port pin signals are modified. If some pins are configured as input or as interrupt, the values are ignored.

7.4 Interrupt Mode

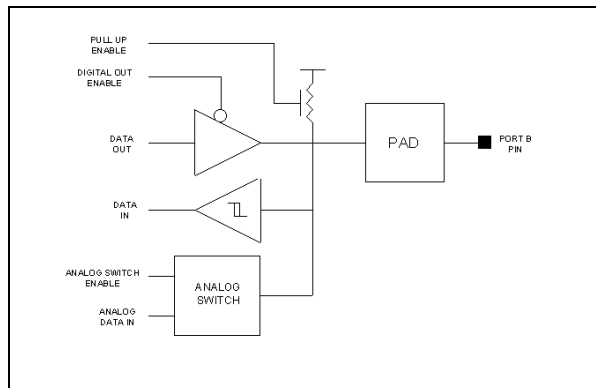
The pins configured as Interrupt Mode can generate a Port Interrupt request. Only Port A and Port B pins can be configured in this mode.

An Interrupt vector is associated to each Port: there are two Port Interrupts available but more pins of the ports can act as source at the same time.

The Configuration Registers switch the signals deriving from interrupt pins to an OR gate that generates the interrupt request signal. The signal deriving from the pins can be read, allowing the discrimination of the interrupt sources when more than one pin can generate the interrupt signal.

The interrupt trigger can be configured either in the rising or falling edge of the external signal.

Figure 7.2 Analog Pin





7.6.1 Configuration Registers.

**Port A Pull-Up Register (PORT\_A\_PULLUP)**

Configuration Register 24 (018h) Read/Write

Reset Value: 0000 0000 (00h)

7							0
PUA7	PUA6	PUA5	PUA4	PUA3	PUA2	PUA1	PUA0

Bit 7-0: **PUA7-0** Port A pull-up (see Table 7.1)

0: Port A pin without pull-up

1: Port A pin with pull-up

**Port A Option Register (PORT\_A\_OR)**

Configuration Register 25 (019h) Read/Write

Reset Value: 0000 0000 (00h)

7							0
ORA7	ORA6	ORA5	ORA4	ORA3	ORA2	ORA1	ORA0

Bit 7-0: **ORA7-0** Port A option (see Table 7.1)

**Port A Data Direction Register (PORT\_A\_DDR)**

Configuration Register 26 (01Ah) Read/Write

Reset Value: 0000 0000 (00h)

7							0
DDRA7	DDRA6	DDRA5	DDRA4	DDRA3	DDRA2	DDRA1	DDRA0

Bit 7-0: **DDRA7-0** Port A direction (see Table 7.1)

0: Port A pin configured as input

1: Port A pin configured as output

**Port A Alternate Fuction (PORT\_A\_AF)**

Configuration Register 27 (01Bh) Read/Write

Reset Value: 0000 0000 (00h)

7							0
AFA7	AFA6	AFA5	AFA4	AFA3	AFA2	AFA1	AFA0

Bit 7: **AFA7** Alternate Function PA7

0: Digital I/O

1: INT

Bit 6: **AFA6** Alternate Function PA6

0: Digital I/O

1: T0OUT

Bit 5: **AFA5** Alternate Function PA5

0: Digital I/O

1: TCLK

Bit 4: **AFA4** Alternate Function PA4

0: Digital I/O

1: TSTRT

Bit 3: **AFA3** Alternate Function PA3

0: Digital I/O

1: TRES

Bit 2: **AFA2** Alternate Function PA2

0: Digital I/O

1: T1OUT

Bit 1: **AFA1** Alternate Function PA1

0: Digital I/O

1: SDA

Bit 0: **AFA0** Alternate Function PA0

0: Digital I/O

1: SCL

**Table 7.1 Pin mode configuration**

MODE	PU	OR	DDR
Input high impedance	0	0	0
Input with pull-up	1	0	0
Interrupt without pull-up	0	1	0
Interrupt with pull-up	1	1	0
Output push-pull	0	0	1
Output with pull-up	1	0	1
Output open drain	0	1	1
Output weak pull-up	1	1	1

**Port B Pull-Up Register (PORT\_B\_PULLUP)**

Configuration Register 28 (01Ch) Read/Write  
Reset Value: 0000 0000 (00h)

7	0						
PUB7*	PUB6*	PUB5**	PUB4**	PUB3**	PUB2**	PUB1	PUB0

(\*) Not used in 16/20 pin package devices  
(\*\*) Not used in 16 pin package devices

Bit 7-0: **PUB7-0** Port B pull-up (see Table 7.1)  
0: Port B pin without pull-up  
1: Port B pin with pull-up

**Port B Option Register (PORT\_B\_OR)**

Configuration Register 29 (01Dh) Read/Write  
Reset Value: 0000 0000 (00h)

7	0						
ORB7*	ORB6*	ORB5**	ORB4**	ORB3**	ORB2**	ORB1	ORB0

(\*) Not used in 16/20 pin package devices  
(\*\*) Not used in 16 pin package devices

Bit 7-0: **ORB7-0** Port B option (see Table 7.1)

**Port B Data Direction Register (PORT\_B\_DDR)**

Configuration Register 30 (01Eh) Read/Write  
Reset Value: 0000 0000 (00h)

7	0						
DDR7*	DDR6*	DDR5**	DDR4**	DDR3**	DDR2**	DDR1	DDR0

(\*) Not used in 16/20 pin package devices  
(\*\*) Not used in 16 pin package devices

Bit 7-0: **DDR7-0** Port B direction (see Table 7.1)  
0: Port B pin configured as input  
1: Port B pin configured as output

**Port B Alternate Function (PORT\_B\_AF)**

Configuration Register 31 (01Fh) Read/Write  
Reset Value: 0000 0000 (00h)

7	0						
AFB7	AFB6	AFB5	AFB4	AFB3	AFB2	AFB1	AFB0

**Note:** This register is not used in 16 pin devices

Bit 7: **AFB7** Alternate Function PB7  
0: Digital I/O  
1: AIN7

Bit 6: **AFB6** Alternate Function PB6  
0: Digital I/O  
1: AIN6

Bit 5: **AFB5** Alternate Function PB5  
0: Digital I/O  
1: AIN5

Bit 4: **AFB4** Alternate Function PB4  
0: Digital I/O  
1: AIN4

Bit 3: **AFB3** Alternate Function PB3  
0: Digital I/O  
1: AIN3

Bit 2: **AFB2** Alternate Function PB2  
0: Digital I/O  
1: AIN2

Bit 1: **AFB1** Alternate Function PB1  
0: Digital I/O  
1: AIN1

Bit 0: **AFB0** Alternate Function PB0  
0: Digital I/O  
1: AIN0 / VREF

**Port C Pull-Up Register (PORT\_C\_PULLUP)**

Configuration Register 32 (020h) Read/Write  
Reset Value: 0000 0000 (00h)

7	0						
-	-	PUC5	PUC4	PUC3	PUC2	PUC1	PUC0

**Note:** This register is not used in 16/20 pin devices

Bit 7-6: Not Used

Bit 5-0: **PUC5-0** Port C pull-up (see Table 7.1)  
0: Port C pin without pull-up

1: Port C pin with pull-up

**Port C Option Register (PORT\_C\_OR)**

Configuration Register 33 (021h) Read/Write  
Reset Value: 0000 0000 (00h)

7	0						
-	-	ORC5	ORC4	ORC3	ORC2	ORC1	ORC0

**Note:** This register is not used in 16/20 pin devices

Bit 7-6: Not Used

Bit 5-0: **ORC5-0** Port C option (see Table 7.1)

**Port C Data Direction Register (PORT\_C\_DDR)**

Configuration Register 34 (022h) Read/Write  
Reset Value: 0000 0000 (00h)

7	0						
-	-	DDRC5	DDRC4	DDRC3	DDRC2	DDRC1	DDRC0

**Note:** This register is not used in 16/20 pin devices

Bit 7-6: Not Used

Bit 5-0: **DDRC5-0** Port C direction (see Table 7.1)  
0: Port C pin configured as input  
1: Port C pin configured as output

**Note:** in order to achieve low current consumption, the port pins must be configured as input pull-up, even though they are not existing in the package. For example in 20 pin devices, the pins PB6-7 and PC0-7 must be configured in input pull-up.

**Port C Alternate Fuction (PORT\_C\_AF)**

Configuration Register 35 (023h) Read/Write  
Reset Value: 0000 0000 (00h)

7	0						
-	-	AFC5	AFC4	AFC3	AFC2	AFC1	AFC0

Bit 7-6: Not Used

Bit 5: **AFC5** Alternate Function PC5  
0: Digital I/O

1: TRES

Bit 4: **AFC4** Alternate Function PC4  
0: Digital I/O  
1: TX

Bit 3: **AFC3** Alternate Function PC3  
0: Digital I/O  
1: SS

Bit 2: **AFC2** Alternate Function PC2  
0: Digital I/O  
1: MISO

Bit 1: **AFC1** Alternate Function PC1  
0: Digital I/O  
1: MOSI

Bit 0: **AFC0** Alternate Function PC0  
0: Digital I/O  
1: SCK

**7.6.2 Input Registers.**

**Port A Data Input Register (PORT\_A\_IN)**

Input Register 0 (00h) Read only  
Reset Value: XXXX XXXX

7	0						
PAI7	PAI6	PAI5	PAI4	PAI3	PAI2	PAI1	PAI0

Bit 7-0: **PAI7-0** Port A Input data

The logical level applied in the Port A pins, configured as digital input, can be achieved by reading this register.

**Port B Data Input Register (PORT\_B\_IN)**

Input Register 1 (01h) Read only  
Reset Value: XXXX XXXX

7	0						
PBI7*	PBI6*	PBI5**	PBI4**	PBI3**	PBI2**	PBI1	PBI0

(\*) Not used in 16/20 pin package devices

(\*\*) Not used in 16 pin package devices



Bit 7-0: **PBI7-0** Port B Input data

The logical level applied in the Port B pins, configured as digital input, can be achieved by reading this register.

**Port C Data Input Register (PORT\_C\_IN)**

Input Register 2 (02h) Read only

Reset Value: XXXX XXXX

7							0
-	-	PCI5	PCI4	PCI3	PCI2	PCI1	PCI0

**Note:** This register is not used in 16/20 pin devices

Bit 7-6: Not Used

Bit 5-0: **PCI5-0** Port C Input data

The logical level applied in the Port C pins, configured as digital input, can be achieved by reading this register.

**7.6.3 Output Registers.**

**Port A Data Output Register (PORT\_A\_OUT)**

Output Register 0 (00h) Write only

Reset Value: 0000 0000 (00h)

7							0
PAO7	PAO6	PAO5	PAO4	PAO3	PAO2	PAO1	PAO0

Bit 7-0: **PAO7-0** Port A Output data

The logical values written in these register bits are put in the Port A pins configured as digital output.

**Port B Data Output Register (PORT\_B\_OUT)**

Output Register 1 (01h) Write only

Reset Value: 0000 0000 (00h)

7						0	
PBO7*	PBO6*	PBO5**	PBO4**	PBO3**	PBO2**	PBO1	PBO0

(\*) Not used in 16/20 pin package devices  
 (\*\*) Not used in 16 pin package devices

Bit 7-0: **PBO7-0** Port B Input data

The logical values written in these register bits are put in the Port B pins configured as digital output.

**Port C Data Output Register (PORT\_C\_OUT)**

Output Register 2 (02h) Write only

Reset Value: 0000 0000 (00h)

7						0	
-	-	PCO5	PCO4	PCO3	PCO2	PCO1	PCO0

**Note:** This register is not used in 16/20 pin devices

Bit 7-6: Not Used

Bit 5-0: **PCO5-0** Port C Input data

The logical values written in these register bits are put in the Port C pins configured as digital output.

### 8 FUZZY COMPUTATION (DP)

The ST52F510/F513/F514 Decision Processor (DP) main features are:

- Up to 8 Inputs with 8-bit resolution;
- 1 Kbyte of Program/Data Memory available to store more than 300 to Membership Functions (Mbf's) for each Input;
- Up to 128 Outputs with 8-bit resolution;
- Possibility of processing fuzzy rules with an UNLIMITED number of antecedents;
- UNLIMITED number of Rules and Fuzzy Blocks.

The limits on the number of Fuzzy Rules and Fuzzy program blocks are only related to the Program/Data Memory size.

#### 8.1 Fuzzy Inference

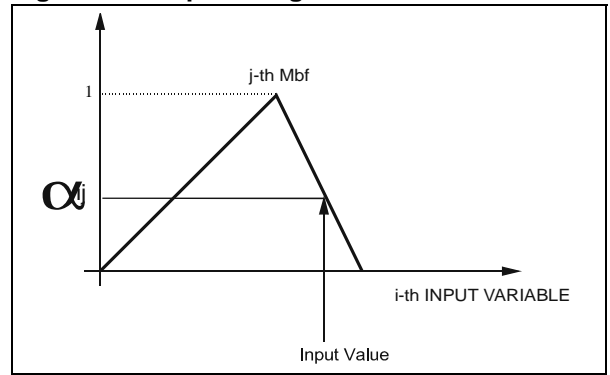
The block diagram shown in Figure 8.1 describes the different steps performed during a Fuzzy algorithm. The ST52F510/F513/F514 Core allows for the implementation of a Mamdani type fuzzy inference with crisp consequents. Inputs for fuzzy inference are stored in 8 dedicated Fuzzy input registers. The LDFR instruction is used to set the Input Fuzzy registers with values stored in the Register File. The result of a Fuzzy inference is stored directly in a location of the Register File.

#### 8.2 Fuzzyfication Phase

In this phase the intersection (alpha weight) between the input values and the related Mbfs (Figure 8.2) is performed.

Eight Fuzzy Input registers are available for Fuzzy inferences.

Figure 8.2 Alpha Weight Calculation



After loading the input values by using the LDFR assembler instruction, the user can start the fuzzy inference by using the FUZZY assembler instruction. During fuzzyfication: input data is transformed in the activation level (alpha weight) of the Mbf's.

#### 8.3 Inference Phase

The Inference Phase manages the alpha weights obtained during the fuzzyfication phase to compute the truth value ( $\omega$ ) for each rule.

This is a calculation of the maximum (for the OR operator) and/or minimum (for the AND operator) performed on alpha values according to the logical connectives of Fuzzy Rules.

Several conditions may be linked together by linguistic connectives AND/OR, NOT operators and brackets.

The truth value  $\omega$  and the related output singleton are used by the Defuzzyfication phase, in order to complete the inference calculation.

Figure 8.1 Fuzzy Inference

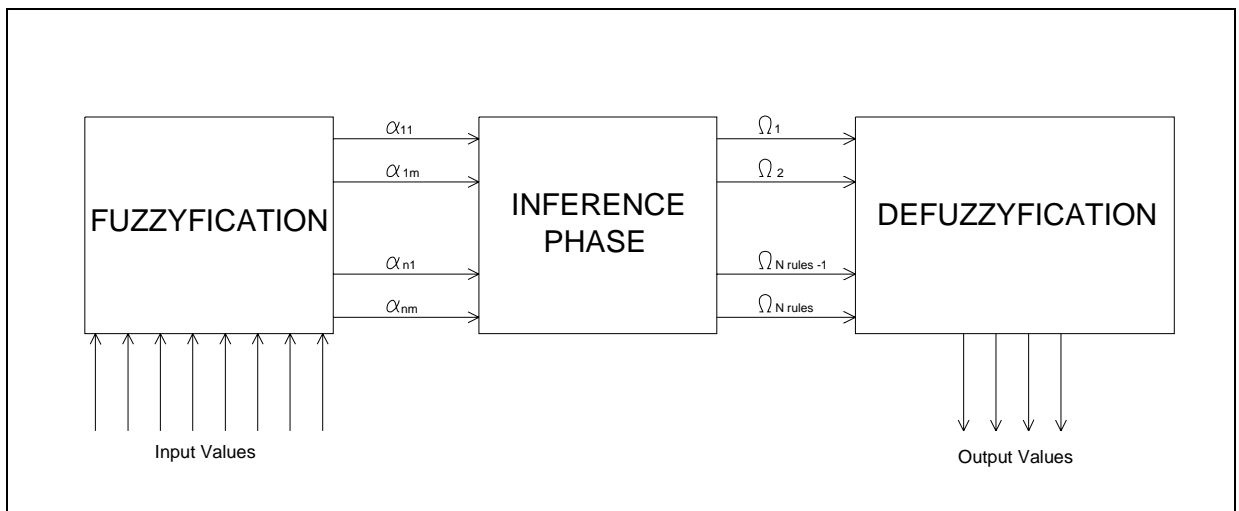
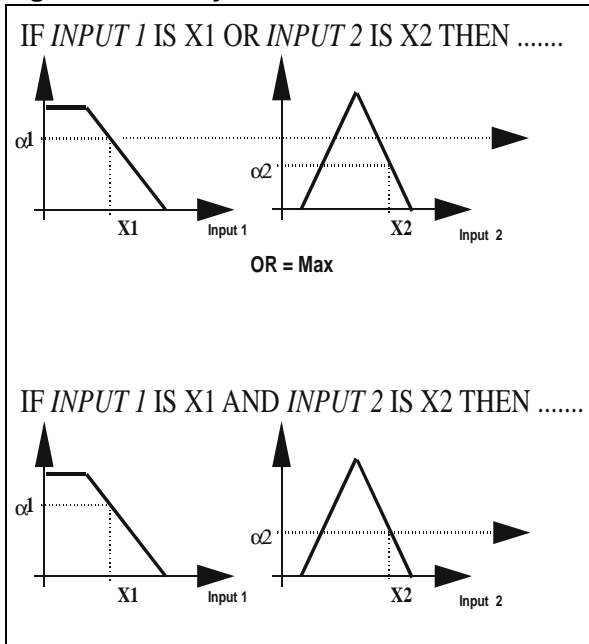




Figure 8.3 Fuzzyfication



### 8.4 Defuzzyfication

In this phase the output crisp values are determined by implementing the consequent part of the rules.

Each consequent Singleton  $X_i$  is multiplied by its weight values  $\omega_i$ , calculated by the Decision processor, in order to compute the upper part of the Defuzzyfication formula.

Each output value is obtained from the consequent crisp values ( $X_i$ ) by carrying out the following Defuzzyfication formula:

$$Y_i = \frac{\sum_j X_{ij} \omega_{ij}}{\sum_j \omega_{ij}}$$

where:

$i$  = identifies the current output variable

$N$  = number of the active rules on the current output

$\omega_{ij}$  = weight of the  $j$ -th singleton

$X_{ij}$  = abscissa of the  $j$ -th singleton

The Decision Processor outputs are stored in the RAM location  $i$ -th specified in the assembler instruction OUT  $i$ .

### 8.5 Input Membership Function

The Decision Processor allows the management of triangular Mbfs. In order to define an Mbf, three different parameters must be stored on the Program/Data Memory (see Figure 8.4):

- the vertex of the Mbf: **V**;
- the length of the left semi-base: **LVD**;
- the length of the right semi-base: **RVD**;

In order to reduce the size of the memory area and the computational effort the vertical range of the vertex is fixed between 0 and 15 (4 bits)

By using the previous memorization method different kinds of triangular Membership Functions may be stored. Figure 8.5 shows some examples of valid Mbfs that can be defined in ST52F510/F513/F514.

Each Mbf is then defined storing 3 bytes in the first Kbyte of the Program/Data Memory.

The Mbf is stored by using the following instruction:

**MBF**  $n\_mbf$   $lvd$   $v$   $rvd$

where:

$n\_mbf$  is a tag number that identifies the Mbf

$lvd$ ,  $v$ , and  $rvd$  are the parameters that describe the Mbf's shape as described above.

Figure 8.4 Mbfs Parameters

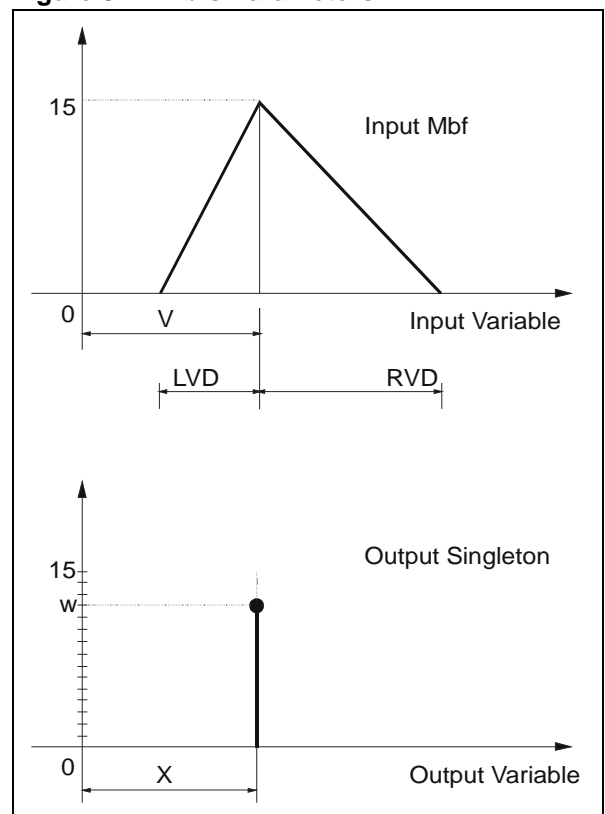
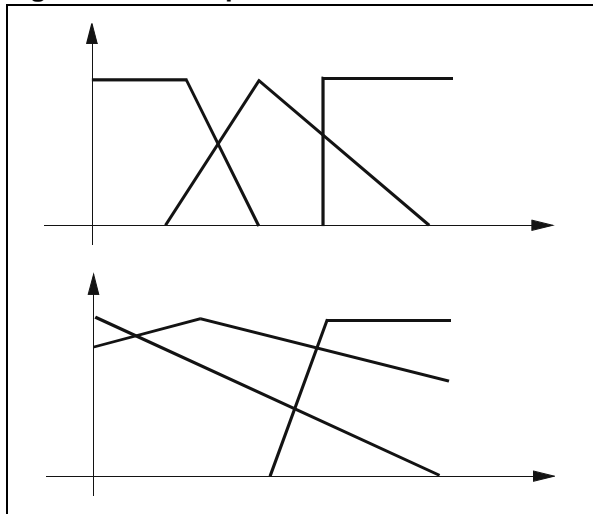


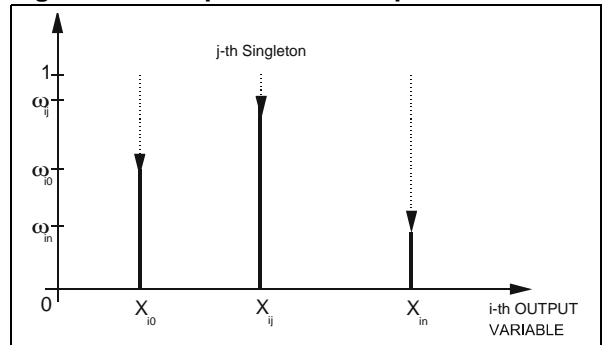
Figure 8.5 Example of valid Mbfs



8.6 Output Singleton

The Decision Processor uses a particular kind of membership function called Singleton for its output variables. A Singleton doesn't have a shape, like a traditional Mbf, and is characterized by a single point identified by the couple (X, w), where w is calculated by the Inference Unit as described earlier. Often, a Singleton is simply identified with its Crisp Value X.

Figure 8.6 Output Membership Functions



8.7 Fuzzy Rules

Rules can have the following structures:

if A op B op C.....then Z

if (A op B) op (C op D op E...) .....then Z

where op is one of the possible linguistic operators (AND/OR)

In the first case the rule operators are managed sequentially; in the second one, the priority of the operator is fixed by the brackets.

Each rule is codified by using an instruction set, the inference time for a rule with 4 antecedents and 1 consequent is about 3 microseconds at 20 MHz.

The Assembler Instruction Set used to manage the Fuzzy operations is reported in the table below.

Table 8.1 Fuzzy Instructions Set

Instruction	Description
<b>MBF</b> <i>n_mbf lvd v rvd</i>	Stores the Mbf <i>n_mbf</i> with the shape identified by the parameters <i>lvd</i> , <i>v</i> and <i>rvd</i>
<b>IS</b> <i>n m</i>	Fixes the alpha value of the input <i>n</i> with the Mbf <i>m</i>
<b>ISNOT</b> <i>n m</i>	Calculates the complementary alpha value of the input <i>n</i> with the Mbf <i>m</i> .
<b>FZAND</b>	Implements the Fuzzy operation AND
<b>FZOR</b>	Implements the Fuzzy operation OR
<b>CON</b> <i>crisp</i>	Multiplies the crisp value with the last $\omega$ weight
<b>OUT</b> <i>n_out</i>	Performs Defuzzyfication and stores the currently Fuzzy output in the register <i>n_out</i>
<b>FUZZY</b>	Starts the computation of a sigle fuzzy variable
( )	Modify the priority in the rule evaluation

**Example 1:**

*IF Input<sub>1</sub> IS NOT Mbf<sub>1</sub> AND Input<sub>4</sub> IS Mbf<sub>12</sub> OR Input<sub>3</sub> IS Mbf<sub>8</sub> THEN Crisp<sub>1</sub>*

is codified by the following instructions:

- ISNOT 1 1** calculates the NOT  $\alpha$  value of Input<sub>1</sub> with Mbf<sub>1</sub> and stores the result in internal registers
- FZAND** implements the operation AND between the previous and the next alpha value evaluated
- IS 4 12** fixes the  $\alpha$  value of Input<sub>4</sub> with Mbf<sub>12</sub> and stores the result in internal registers
- FZOR** implements the operation OR between the previous and the next alpha value evaluated
- IS 3 8** fixes the  $\alpha$  value of Input<sub>3</sub> with Mbf<sub>8</sub> and stores the result in internal registers
- CON crisp<sub>1</sub>** multiplies the result of the last  $\Omega$  operation with the crisp value *crisp<sub>1</sub>*

**Example 2**, the priority of the operator is fixed by the brackets:

*IF (Input<sub>3</sub> IS Mbf<sub>1</sub> AND Input<sub>4</sub> IS NOT Mbf<sub>15</sub>) OR (Input<sub>1</sub> IS Mbf<sub>6</sub> OR Input<sub>6</sub> IS NOT Mbf<sub>14</sub>) THEN Crisp<sub>2</sub>*

- (** parenthesis open to change the priority
- IS 3 1** fixes the  $\alpha$  value of Input<sub>3</sub> with Mbf<sub>1</sub> and stores the result in internal registers
- FZAND** implements the operation AND between the previous and the next alpha value evaluated
- ISNOT 4 15** calculates the NOT  $\alpha$  value of Input<sub>4</sub> with Mbf<sub>15</sub> and stores the result in internal registers
- )** parenthesis closed
- FZOR** implements the operation OR between the previous and the next alpha value evaluated
- (** parenthesis open to change the priority
- IS 1 6** fixes the  $\alpha$  value of Input<sub>1</sub> with Mbf<sub>6</sub> and stores the result in internal registers
- FZOR** implements the operation OR between the previous and the next alpha value evaluated
- ISNOT 2 14** calculates the NOT  $\alpha$  value of Input<sub>6</sub> with Mbf<sub>14</sub> and stores the result in internal registers
- )** parenthesis closed
- CON crisp<sub>2</sub>** multiplies the result of the last  $\Omega$  operation with the crisp value *crisp<sub>2</sub>*

At the end of the fuzzy rules related to the current Fuzzy Variable, by using the instruction **OUT reg**, the specified register is written with the computed value. Afterwards, the control of the algorithm returns to the CU. The next Fuzzy Variable evaluation must start again with a FUZZY instruction.

**9 INSTRUCTION SET**

ST52F510/F513/F514 supplies 107 (98 + 9 Fuzzy) instructions that perform computations and control the device. Computational time required for each instruction consists of one clock pulse for each Cycle plus 2 clock pulses for the decoding phase. Total computation time for each instruction is reported in Table 9.1

The ALU of ST52F510/F513/F514 can perform multiplication (MULT) and division (DIV). Multiplication is performed by using 8 bit operands storing the result in 2 registers (16 bit values), see Figure 2.3.

Division is performed between a 16 bit dividend and an 8 bit divider, the result and the remainder are stored in two 8-bit registers (see Figure 2.4).

**9.1 Addressing Modes**

ST52F510/F513/F514 instructions allow the following addressing modes:

- **Inherent:** this instruction type does not require an operand because the opcode specifies all the information necessary to carry out the instruction. Examples: NOP, SCF.
- **Immediate:** these instructions have an operand as a source immediate value. Examples: LDRC, ADDI.
- **Direct:** the operands of these instructions are specified with the direct addresses. The

operands can refer (according to the opcode) to addresses belonging to the different addressing spaces. Example: SUB, LDRE.

- **Indirect:** data addresses that are required are found in the locations specified as operands. Both source and/or destination operands can be addressed indirectly. The operands can refer, (according to the opcode) to addresses belonging to different addressing spaces. Examples: LDRR(reg1),(reg2); LDER mem\_addr,(reg1).
- **Bit Direct:** operands of these instructions directly address the bits of the specified Register File locations. Examples: BSET, BTEST.

**9.2 Instruction Types**

ST52F510/F513/F514 supplies the following instruction types:

- Load Instructions
- Arithmetic and Logic Instructions
- Bitwise instructions
- Jump Instructions
- Interrupt Management Instructions
- Control Instructions

The instructions are listed in Table 9.1

**Table 9.1 Instruction Set**

Load Instructions						
Mnemonic	Instruction	Bytes	Cycles	Z	S	C
BLKSET	BLKSET const	2	(*)	-	-	-
GETPG	GETPG regx	2	7	-	-	-
LDCE	LDCE confx,memy	3	8/9	-	-	-
LDCI	LDCI confx, const	3	7	-	-	-
LDCNF	LDCNF regx, conf	3	7	-	-	-
LDCR	LDCR confx, regy	3	8	-	-	-
LDER	LDER memx, regy	3	10	-	-	-
LDER	LDER (regx),(regy)	3	11	-	-	-
LDER	LDER (regx), regy	3	10	-	-	-
LDER	LDER memx,(regy)	3	11	-	-	-
LDFR	LDFR fuzzyx, regy	3	8	-	-	-

Load Instructions (continued)						
LDPE	LDPE outx, memy	3	8/9	-	-	-
LDPE	LDPE outx, (regy)	3	9/10	-	-	-
LDPI	LDPI outx, const	3	7	-	-	-
LDPR	LDPR outx, regy	3	8	-	-	-
LDRC	LDRC regx, const	3	7	-	-	-
LDRE	LDRE regx, memy	3	8/9	-	-	-
LDRE	LDRE (regx), (regy)	3	10/11	-	-	-
LDRE	LDRE (regx), memy	3	9/10	-	-	-
LDRE	LDRE regx, (regy)	3	9/10	-	-	-
LDRI	LDRI regx, inpx	3	7	-	-	-
LDRR	LDRR regx, regy	3	9	-	-	-
LDRR	LDRR (regx), (regy)	3	10	-	-	-
LDRR	LDRR (regx), regy	3	9	-	-	-
LDRR	LDRR regx, (regy)	3	10	-	-	-
PGSET	PGSET const	2	4	-	-	-
PGSETR	PGSETR regx	2	5	-	-	-
POP	POP regx	2	7	-	-	-
PUSH	PUSH regx	2	8	-	-	-

Arithmetic Instructions						
Mnemonic	Instruction	Bytes	Cycles	Z	S	C
ADD	ADD regx, regy	3	9		-	
ADDC	ADDC regx, regy	3	9		-	
ADDI	ADDI regx, const	3	8		-	
ADDIC	ADDIC regx, const	3	8		-	
ADDO	ADDO regx, regy	3	11			
ADDOC	ADDOC regx, regy	3	11			
ADDOI	ADDOI regx, const	3	10			
ADDOIC	ADDOIC regx, const	3	10			
AND	AND regx, regy	3	9		-	-
ANDI	ANDI regx, const	3	8		-	-
CP	CP regx, regy	3	8			-
CPI	CPI regx, const	3	7			-
DEC	DEC regx	2	7			-

Arithmetic Instructions (continued)						
DIV	DIV regx, regy	3	16			
INC	INC regx	2	7		-	
MIRROR	MIRROR regx	2	7		-	-
MULT	MULT regx, regy	3	11		-	-
NOT	NOT regx	2	7		-	-
OR	OR regx, regy	3	9		-	-
ORI	ORI regx, const	3	8		-	-
SUB	SUB regx, regy	3	9			-
SUBI	SUBI regx, const	3	8			-
SUBIS	SUBIS regx, const	3	8			-
SUBO	SUBO regx, regy	3	11			
SUBOI	SUBOI regx, const	3	10			
SUBOIS	SUBOISregx,const	3	10			
SUBOS	SUBOS regx, regy	3	11			
SUBS	SUBS regx, regy	3	9			-
RCF	RCF	1	4	-	-	
RSF	RSF	1	4	-		-
RZF	RZF	1	4		-	-
SCF	SCF	1	4	-	-	
SSF	SSF	1	4	-		-
SZF	SZF	1	4		-	-
XOR	XOR regx, regy	3	9		-	-
XORI	XORI regx, cons	3	8		-	-

Bitwise Instructions						
Mnemonic	Instruction	Bytes	Cycles	Z	S	C
ASL	ASL regx	2	7		-	
ASR	ASR regx	2	7			-
BNOT	BNOT regx, bit	3	8		-	-
BRES	BRES regx, bit	3	8		-	-
BSET	BSET regx, bit	3	8		-	-
BTEST	BTEST regx, bit	3	7		-	-
MTEST	MTEST regx,const	3	7		-	-
RLC	RLC regx	2	7		-	

## Bitwise Instructions (continued)

Mnemonic	Instruction	Bytes	Cycles	Z	S	C
ROL	ROL regx	2	7		-	
ROR	ROR regx	2	7			-
RRS	RRS regx	2	7			-

## Jump Instructions

Mnemonic	Instruction	Bytes	Cycles	Z	S	C
CALL	CALL addr	3	11	-	-	-
JP	JP addr	3	6	-	-	-
JPC	JPC addr	3	5/6	-	-	-
JPNC	JPNC addr	3	5/6	-	-	-
JPNS	JPNS addr	3	5/6	-	-	-
JPNZ	JPNZ addr	3	5/6	-	-	-
JPS	JPS addr	3	5/6	-	-	-
JPZ	JPZ addr	3	5/6	-	-	-
RET	RET	1	8	-	-	-

## Interrupt Management Instructions

Mnemonic	Instruction	Bytes	Cycles	Z	S	C
HALT	HALT	1	4/13	-	-	-
MEGI	MEGI	1	6/11	-	-	-
MDGI	MDGI	1	5	-	-	-
RETI	RETI	1	9	-	-	-
RINT	RINT INT	2	6	-	-	-
UDGI	UDGI	1	5	-	-	-
UEGI	UEGI	1	6/11	-	-	-
TRAP	TRAP	1	9	-	-	-
WAITI	WAITI	1	7/10	-	-	-

## Control Instructions

Mnemonic	Instruction	Bytes	Cycles	Z	S	C
FUZZY	FUZZY	1	4	-	-	-
NOP	NOP	1	5	-	-	-
WDTRFR	WDTRFR	1	6	-	-	-
WDTSLP	WDTSLP	1	5	-	-	-

**Notes:**

regx, regy:	Register File Address
memx, memy:	Program/Data Memory Addresses
confx, confy:	Configuration Registers Addresses
outx:	Output Registers Addresses
inpx:	Input Registers Addresses
const:	Constant value
fuzzyx:	Fuzzy Input Registers
	flag affected
-	flag not affected

(\*) The instruction BLKSET determines the start of a 32 byte block writing in Flash or EEPROM Program/Data Memory. During this phase (about 4 ms), the CPU is stopped to executing program instructions. The duration of the BLKSET instruction can be identified with this time.



## 10 10-BIT A/D CONVERTER

### 10.1 Introduction

ST52F510/F513/F514 A/D Converter is a 10-bit analog to digital converter with up to 8 analog inputs. The A/D converter offers a typical conversion time of 10  $\mu$ s in fast mode and of 20  $\mu$ s in slow mode. This period also includes the time of the integral Sample and Hold circuitry, which minimizes the need for external components and allows quick sampling of the signal for the minimum warping effect and integral conversion error.

In addition the peripheral performs a calibration procedure in order to get the maximum precision allowed in the data of conversion. The calibration procedure is performed in two phases: the pre-charging phase and the tuning phase. The pre-charging process can be executed, after the peripheral start, to set-up the internal references and to speed-up the tuning process. The tuning process is carried-out during the channels conversion.

**Note:** The user must be take in account both the pre-charging time and some dummy conversion (at least 20) for the tuning before starting the data acquisition. It is recommended to repeat this procedure at the start-up and after a long time peripheral stop.

The pre-charging process starts by starting the peripheral by setting to 1 the STR bit of the AD\_CR Configuration Register. To speed-up the calibration procedure, the pre-charging phase can be skipped when not necessary (for example when consecutive single conversions are performed). The user can disable the pre-charging by setting the PRECH bit in the AD\_CR Configuration Register.

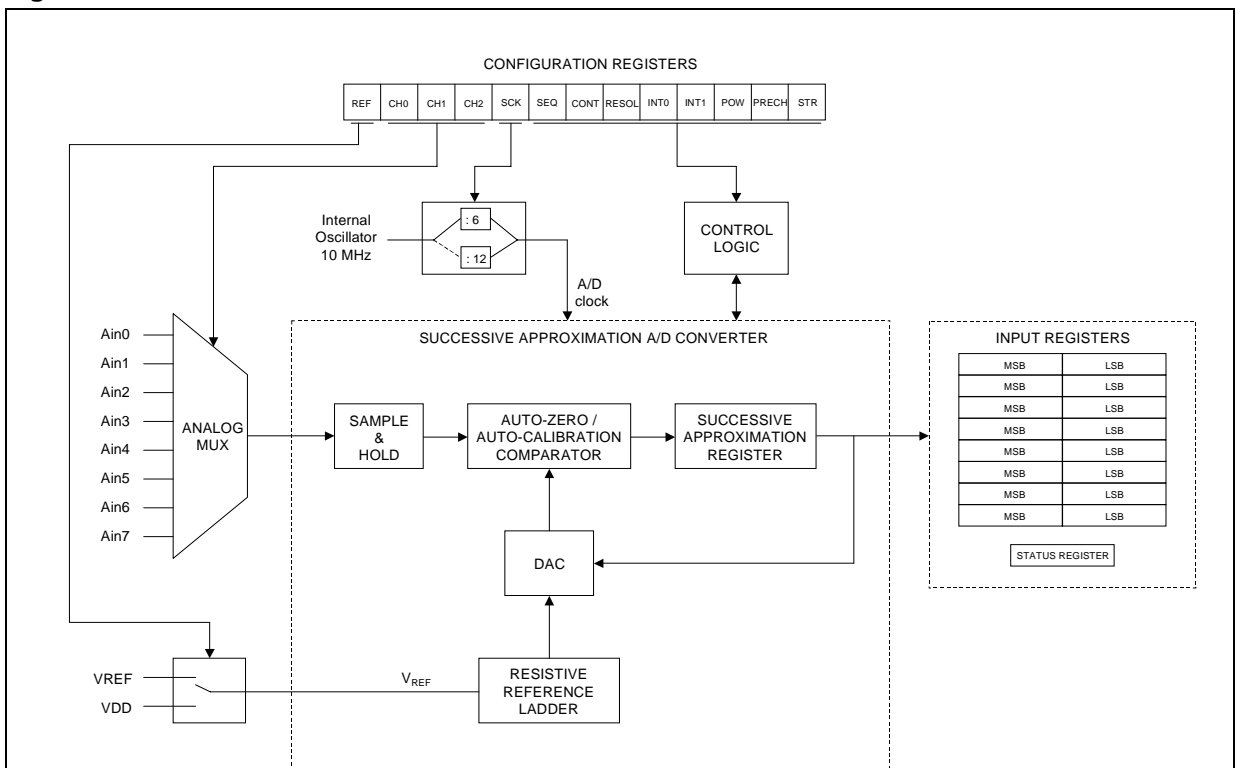
The A/D peripheral converts the input voltage with a process of successive approximations using a fixed clock frequency derived from the 10 MHz internal oscillator, divided by a factor that depends on the speed mode: about 1.6 MHz in Fast Mode and 800 kHz in Slow Mode. The speed mode is chosen by the SCK bit of the AD\_CR Configuration Register.

The conversion range is found between the analog  $V_{SS}$  and the A/D  $V_{REF}$  references. The  $V_{REF}$  can be either internal, derived from the  $V_{DD}$ , or external by using the VREF pin. The external reference voltage allows the application of more precise and stable reference voltages. The two modes are selected by using the REF bit of the AD\_CR Configuration Register.

**Remark:** the voltage applied to the VREF pin must be in the range  $V_{SS}$ - $V_{DD}$ .

The external reference voltage  $V_{REF}$  is applied to the analog pin PB0. This pin shares the alternate functions with the first analog channel Ain0: if the

Figure 10.1 A/D Converter Structure



external reference mode is chosen the Ain0 channel is not used and the first channel of the conversion sequence becomes Ain1.

The converter uses a fully differential analog input configuration for a better noise immunity and precision performances.

Up to 8 multiplexed Analog Inputs are available. A single signal or a group of signals can be converted sequentially by simply programming the starting address of the last analog channel to be converted. Single or continuous conversion modes are available.

The result of the conversion of each A/D channel is stored in the 8-bit Input Register pairs (addresses from 41 to 56 (029h-038h)) according to the 8-bit or 10-bit mode. The resolution of conversion (8 or 10 bit) can be chosen by programming the RESOL bit of the AD\_CR Configuration Register. In 8-bit mode the eight most significative bits (9:2) of the result of conversion is stored in the least significative byte of the register pair and the most significative is put to zero. In 10-bit mode the two most significative bits (9:8) are stored in the most significative byte of the register pair; the other bits (7:0) are stored in the least significative byte.

In 10-bit mode the result of the conversion must be read in two steps: the MSB and the LSB. The peripheral has been designed to avoid the side effects that can occur when the register are modified between the reading of the two bytes. In fact the latching of the input register pair is disabled after the reading of the first byte and it is enabled again after the reading of the second byte. User should pay attention to complete the two readings to guarantee the data of the conversion to be latched.

When the converted signal is higher than  $V_{REF}$ , an overflow occurs. In this case the 8/10 bits result are all set to 1 and the A/D Overflow Register bit (address 39 027h) corresponding to the channel is set to 1. The bit is reset at the next conversion having no overflow occurrence.

ST52F510/F513/F514 Interrupt Unit provides one maskable channel for the End of Conversion and for the overflow control. It is possible to set the interrupt source on EOC or on overflow or on both by programming the INT0 and INT1 bits in the AD\_CR Configuration Registers.

**Note:** the A/D Converter interrupts are not enabled unless the bit 0 (MSKAD) of the Configuration Register 0 (INT\_MASK) is enabled (set to 1).

A Power-Down programmable bit (POW) allows the A/D converter to be set to a minimum consumption idle status. A stabilization time is required, after the Power On, before accurate conversions can be performed.

## 10.2 Functional Description

The conversion is monotonic, meaning that the result never decreases if the analog input doesn't and never increases if the analog input doesn't.

If input voltage is less than  $V_{SS}$  (voltage supply low) then the result is equal to 00h.

The A/D converter is linear and the digital result of the conversion is provided by the following formula:

$$Digitalresult = \frac{255 \times InputVoltage}{ReferenceVoltage}$$

Where Reference Voltage is  $V_{ref} - V_{SS}$ .

The accuracy of the conversion is described in the Electrical Characteristics Section of the device datasheets.

The A/D converter is not affected by the WAIT mode.

When the ICU enters HALT mode with the A/D converter enabled, the converter is disabled until HALT mode is exited and the start-up delay has elapsed.

## 10.3 Operating Modes

Four main operating modes can be selected by setting the values of the CONT and SEQ bit in the A/D Configuration Register AD\_CR.

**10.3.1 One Channel Single Mode.** In this mode (CONT=0, SEQ=0), the A/D provides an EOC signal after the end of the conversion of the specified channel; then the A/D waits for a new start event. The channel is identified by the bits CH2-CH0 in the Configuration Register AD\_CR, while the bit STR is used to command the Start/Stop.

**10.3.2 Multiple Channels Single Mode.** In this mode (CONT=0, SEQ=1) the A/D provides an EOC signal after the end of the channels sequence conversion identified by the three AD\_CR Configuration Register bits CH2-0; then A/D waits for a new start event.

**10.3.3 One Channel Continuous Mode.** In this mode (CONT=1, SEQ=0) a continuous conversion flow is entered by a start event on the selected channel. At the end of each conversion, the relative Input Register is updated with the last conversion result, while the former value is lost.

The conversion continues until a stop command is executed by writing a '0' in the apposite AD\_CR Configuration Register bit STR.

### 10.3.4 Multiple Channels Continuous Mode.

In this mode (CONT=1, SEQ=1) a continuous conversion flow is entered by a start event on the selected channel sequence. The CH2-0 bits indicate the last channel of the sequence.

At the end of each conversion the relative Input Registers are updated with the last conversion results, while the former values are lost.

The conversion continues until a stop command is executed by writing a '0' in the apposite AD\_CR Configuration Register bit STR.

### 10.4 Power Down Mode

Before enabling any A/D operation modes, set the Power On bit (POW) of the Configuration Register AD\_CR to '1' and then start the A/D Converter by setting the STR bit. It is suggested to execute the pre-charging after the Power on to speed-up the auto calibration process. Clearing the Power On bit is useful when the A/D is not used, reducing the total chip power consumption. This state is also the reset configuration and it is forced by hardware when the core is in HALT state (after a HALT instruction execution).

### 10.5 A/D Converter Register Description

The following registers are related to the use of the A/D Converter.

#### 10.5.1 A/D Converter Configuration Registers.

##### A/D Converter Control Register 1 (AD\_CR1)

Configuration Register 8 (08h) Read/Write  
Reset Value: 0000 0000 (00h)

7							0
CH2	CH1	CH0	SCK	SEQ	POW	CONT	STR

Bit 7-5: **CH2-CH0** Channel Number

The number specified identifies the number of channels to be converted (Multiple Channel mode) or the channel to be converted (One Channel mode)

Bit 4: **SCK** A/D speed mode

- 0: Slow mode (800 kHz)
- 1: Fast mode (1600 kHz)

Bit 3: **SEQ** One/Multiple Channel Mode

- 0: One Channel Mode
- 1: Multiple Channel Mode

Bit 2: **POW** A/D Converter Power Down/Up

- 0: Power down
- 1: Power up

Bit 1: **CONT** Single/Continuous Mode

- 0: Single Mode
- 1: Continuous Mode

Bit 0: **STR** A/D Converter Start bit

- 0: A/D Converter stopped
- 1: A/D Converter started

##### A/D Converter Control Register 2 (AD\_CR2)

Configuration Register 47 (02Fh) Read/Write  
Reset Value: 0000 0000 (00h)

7								0
-	-	-	PRECH	REF	RESOL	INT1	INT0	

Bit 7-5: not used

Bit 4: **PRECH** Pre-charging process on/off

- 0: Pre-charge on (default)
- 1: Pre-charge off

Bit 3: **REF** Voltage Reference (VREF) source

- 0: Internal from Vdd
- 1: External from VREF pin

Bit 2: **RESOL** 8/10 bits resolution

- 0: 10 bits
- 1: 8 bits

Bit 1: **INT1** Overflow interrupt mask

- 0: interrupt disabled
- 1: interrupt enabled (if MSKAD=1)

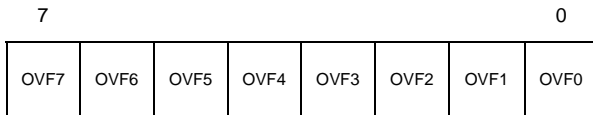
Bit 0: **INT0** End of Conversion interrupt mask

- 0: interrupt disabled
- 1: interrupt enabled (if MSKAD=1)

**10.5.2 Input Registers.**

**A/D Converter Overflow Register (AD\_OVF)**

Input Register 39 (027h) Read only  
 Reset Value: 0000 0000 (00h)



Bit 7-0: **OVF7-OVF0** Overflow Flag  
 0: no overflow occurred in the last conversion  
 1: overflow occurred in the last conversion

**A/D Converter Data Registers**

The converted digital values of the analog level applied to AIN0-7 pins, are buffered in the following register couples:

**A/D Channel 0 data MSB (CHAN0\_H)**

Input Register 41 (029h) Read only  
 Reset Value: 0000 0000 (00h)

**A/D Channel 0 data LSB (CHAN0\_L)**

Input Register 42 (02Ah) Read only  
 Reset Value: 0000 0000 (00h)

**A/D Channel 1 data MSB (CHAN1\_H)**

Input Register 43 (02Bh) Read only  
 Reset Value: 0000 0000 (00h)

**A/D Channel 1 data LSB (CHAN1\_L)**

Input Register 44 (02Ch) Read only  
 Reset Value: 0000 0000 (00h)

**A/D Channel 2 data MSB (CHAN2\_H)**

Input Register 45 (02Dh) Read only  
 Reset Value: 0000 0000 (00h)

**A/D Channel 2 data LSB (CHAN2\_L)**

Input Register 46 (02Eh) Read only

Reset Value: 0000 0000 (00h)

**A/D Channel 3 data MSB (CHAN3\_H)**

Input Register 47 (02Fh) Read only  
 Reset Value: 0000 0000 (00h)

**A/D Channel 3 data LSB (CHAN3\_L)**

Input Register 48 (030h) Read only  
 Reset Value: 0000 0000 (00h)

**A/D Channel 4 data MSB (CHAN4\_H)**

Input Register 49 (031h) Read only  
 Reset Value: 0000 0000 (00h)

**A/D Channel 4 data LSB (CHAN4\_L)**

Input Register 50 (032h) Read only  
 Reset Value: 0000 0000 (00h)

**A/D Channel 5 data MSB (CHAN5\_H)**

Input Register 51 (033h) Read only  
 Reset Value: 0000 0000 (00h)

**A/D Channel 5 data LSB (CHAN5\_L)**

Input Register 52 (034h) Read only  
 Reset Value: 0000 0000 (00h)

**A/D Channel 6 data MSB (CHAN6\_H)**

Input Register 53 (035h) Read only  
 Reset Value: 0000 0000 (00h)

**A/D Channel 6 data LSB (CHAN6\_L)**

Input Register 54 (036h) Read only  
 Reset Value: 0000 0000 (00h)

**A/D Channel 7 data MSB (CHAN7\_H)**

Input Register 55 (037h) Read only  
 Reset Value: 0000 0000 (00h)

**A/D Channel 7 data LSB (CHAN7\_L)**

Input Register 56 (038h) Read only  
 Reset Value: 0000 0000 (00h)

## 11 WATCHDOG TIMER

### 11.1 Functional Description

The Watchdog Timer (WDT) is used to detect the occurrence of a software fault, usually generated by external interference or by unforeseen logical conditions, which causes the application program to abandon its normal sequence. The WDT circuit generates an ICU reset on expiry of a programmed time period, unless the program refreshes the WDT before the end of the programmed time delay. Sixteen different delays can be selected by using the WDT configuration register.

After the end of the delay programmed by the configuration register, if the WDT is active, it starts a reset cycle pulling the reset signal low.

Once the WDT is activated, the application program has to refresh the counter (by the WDTRFR instruction) during normal operation in order to prevent an ICU reset.

In ST52F510/F513/F514 devices it is possible to choose between "Hardware" or "Software" Watchdog. The Hardware WDT allows the counting to avoid unwanted stops for external interferences. The first mode is always enabled unless the Option Byte 4 (WDT\_EN) is written with a special code (10101010b): only this code can switch the WDT in "Software" Mode, the other 255 possibilities keep the "Hardware" Mode enabled.

The WDT is started and refreshed by using the WDTRFR instruction. When the software mode is enabled, the WDTSLP instruction stops the WDT avoiding timeout resets.

When the WDT is in Hardware Mode, neither the WDTSLP instruction nor external interference can stop the counting. The "Hardware" WDT is always enabled after a Reset.

The working frequency of WDT (PRES CLK in the Figure 11.1) is equal to the clock master. The clock master is divided by 500, obtaining the WDT CLK signal that is used to fix the timeout of the WDT.

According to the WDT\_CR Configuration Register values, a WDT delay between 0.1ms and 937.5ms can be defined when the clock master is 5 MHz. By changing the clock master frequency the timeout delay can be calculated according to the configuration register values. The first 4 bits of the WDT\_CR register are used, obtaining 16 different delays.

**Table 11.1 Watchdog Timing Range (5 MHz)**

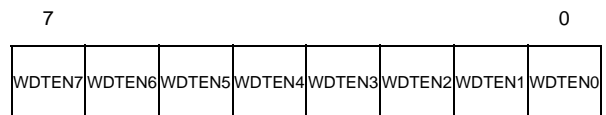
WDT timeout period (ms)	
min	0.1
max	937.5

### 11.2 Register Description

#### SW Watchdog Enable (WDT\_EN)

Option Byte 4 (04h)

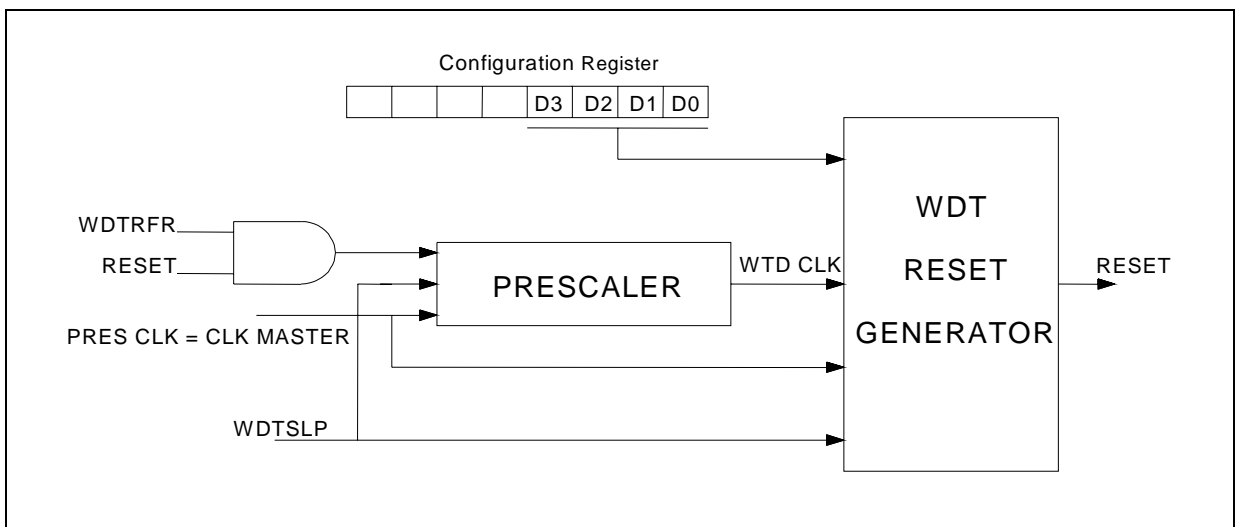
Reset Value: 0000 0000 (00h)



Bit 7-0: **WDTEN7-0** SW Watchdog Enable byte

Writing the code 10101010 in this byte the Software Watchdog mode is enabled.

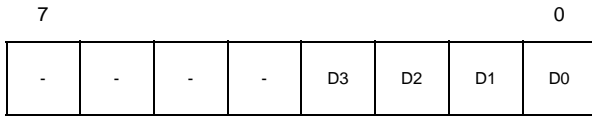
**Figure 11.1 Watchdog Block Diagram**



**Watchdog Control Register (WDT\_CR)**

Configuration Register 7 (07h) Read/Write

Reset Value: 0000 0001 (00h)



Bit 3-0: **D3-0** Watchdog Clock divisor factor bits

The Watchdog Clock (WDT CLK) is divided by the numeric factor determined by these bits, according with Table 11.2 and the following formula:

$$Timeout(ms) = \frac{5 \times 10^5 \times DivisionFactor}{Clock(MHz)}$$

Bit 7-4: Not Used

**Table 11.2 Watchdog Timeout configuration examples**

WDT_CR(3:0)	Division Factor	Timeout Values (ms)		
		5 MHz	10 MHz	20MHz
0000	1	0.1	0.05	0.025
0001	625	62.5	31.25	15.625
0010	1250	125	62.5	31.25
0011	1875	187.5	93.75	46.875
0100	2500	250	125	62.5
0101	3125	312.5	156.25	78.125
0110	3750	375	187.5	93.75
0111	4375	437.5	218.75	109.375
1000	5000	500	250	125
1001	5625	562.5	281.25	140.625
1010	6250	625	312.5	156.25
1011	6875	687.5	343.75	171.875
1100	7500	750	375	187.5
1101	8125	812.5	406.25	203.125
1110	8750	875	437.5	218.75
1111	9375	937.5	468.75	234.375

## 12 PWM/TIMERS

### 12.1 Introduction

ST52F510/513/514 offers two on-chip PWM/Timer peripherals. All ST52F510/513/514 PWM/Timers have the same internal structure. The timer consists of a 16-bit counter with a 16-bit programmable Prescaler, giving a maximum count of  $2^{32}$  (see Figure 12.1).

Each timer has two different working modes, which can be selected by setting the correspondent bit TxMOD of the PWMx\_CR1 Configuration Register: Timer Mode and PWM (Pulse Width Modulation) Mode.

All the Timers have Autoreload Functions; in PWM Mode the reload value can be set by the user.

Each timer output is available on the apposite external pins configured in Alternate Function and in one of the Output modes.

PWM/Timer 0 can also use external START/STOP signals in order to perform Input capture and Output compare, external RESET signal, and external CLOCK to count external events: TSTRT, TRES and TCLK pins. In addition, the START/STOP and RESET signals have configurable polarity (falling or rising edge).

**Remark:** To use TRES, TSTRT, TCLK external signals the related pins must be configured in Alternate Function and in one of Input modes.

For each timer, the contents of the 16-bit counter are incremented on the Rising Edge of the 16-bit prescaler output (PRESCOUT) and it can be read at any instant of the counting phase by accessing the Input Registers PWMx\_COUNT\_IN\_x; the value is stored in two 8-bit registers (MSB and LSB) for each PWM/Timer.

The Input Registers couple PWMx\_CAPTURE\_x store the counter value after the last Stop signal (only Timer Mode). The counter value is not stored after a Reset Signal.

The peripheral status can also be read from the Input Registers (one for each Timer). These registers report START/STOP, SET/RESET status, TxOUT signal and the counter overflow flag. This last signal is set after the first EOC and it is reset by a Timer RESET (internal or external).

### 12.2 Timer Mode

Timer Mode is selected writing 0 in the TxMOD bit.

Each Timer requires three signals: Timer Clock (TMRCLKx), Timer Reset (TxRES) and Timer Start (TxSTRT) (see Figure 12.1). Each of these signals can be generated internally, and/or externally only for Timer 0, by using TRES, TSTRT and TCLK pins.

The Prescaler output (PRESCOUT) increments the Counter value on the rising edge. PRESCOUT is obtained from the internal clock signal (CLKM) or, only for TIMER0, from the external signal provided on the apposite pin.

**Note:** The external clock signal applied on the TCLK pin must have a frequency that is at least two times smaller than the internal master clock.

The prescaler output period can be selected by setting the TxPRESC bits with one of the 17 division factors available. TMRCLK frequency is divided by a factor equal to the power of two of the prescaler values (up to  $2^{16}$ ).

TxRES resets the content of the 16-bit counter to zero. It is generated by writing 0 in the TxRES bit of the PWMx\_CR1 Configuration Register and/or it can be driven by the TRES pin if configured (only Timer0).

Figure 12.1 PWM/Timer Counter block diagram

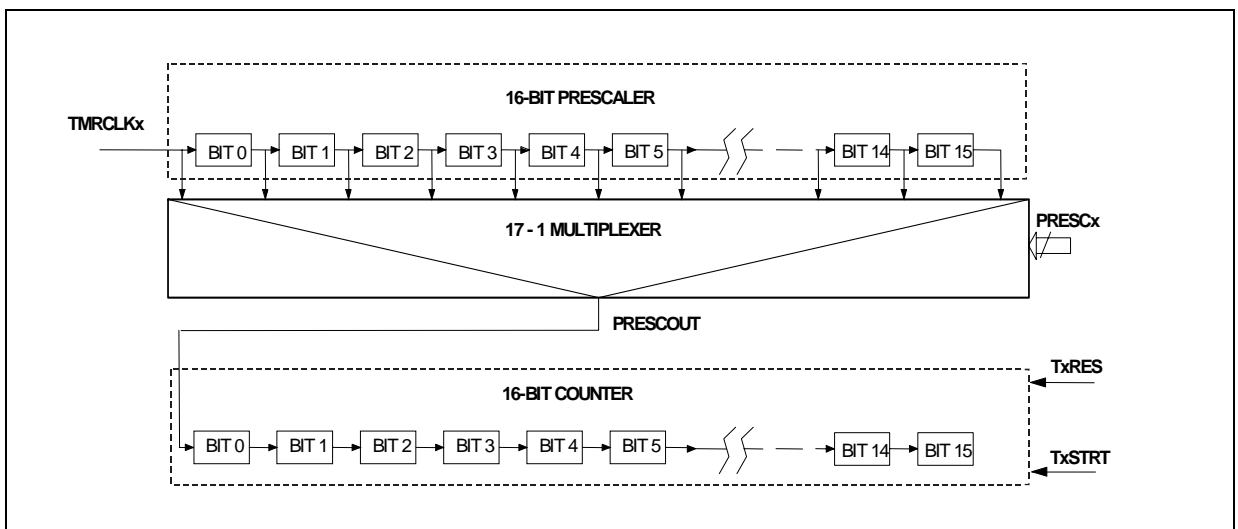
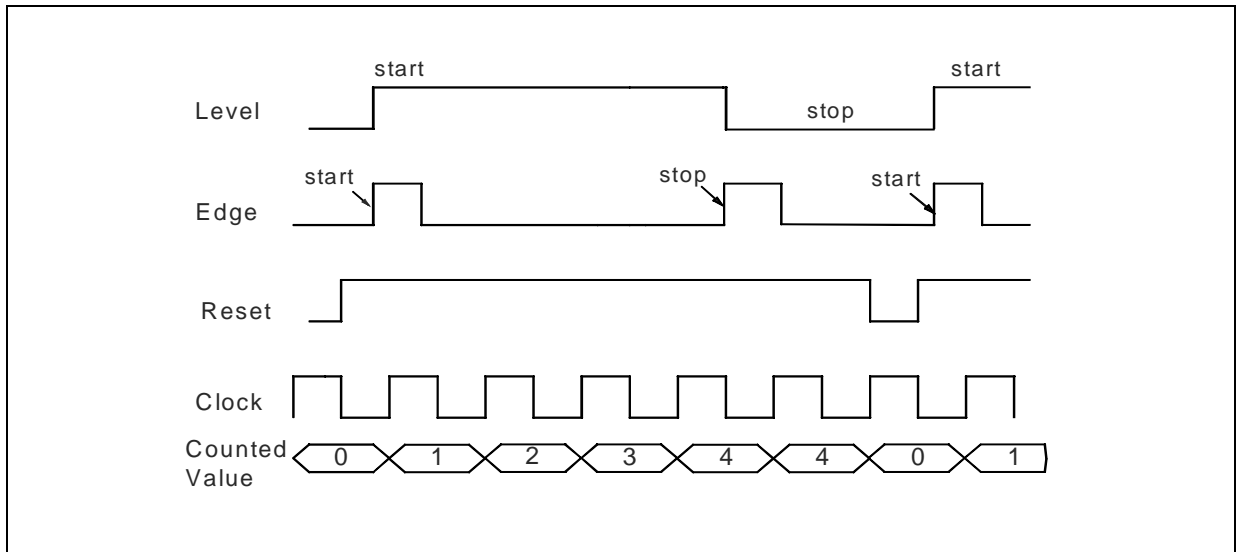


Figure 12.2 Timer 0 External Start/Stop Mode



TxSTRT signal starts/stops the Timer from counting only if the peripherals are configured in Timer mode. The Timers are started by writing 1 in the TXSTRT bit of the PWMx\_CR1 and are stopped by writing 0. This signal can be generated internally and/or externally by forcing the TSTRT pin (only TIMER0).

TIMER 0 START/STOP can be given externally on the TSTRT pin. In this case, the T0STRT signal allows the user to work in two different configurable modes (see Figure 12.2):

- LEVEL (Time Counter): If the T0STRT signal is high, the Timer starts counting. When the T0STRT is low the timer stops counting and the 16-bit current value is stored in the PWM0\_COUNT\_IN\_x Input Registers couple.
- EDGE (Period Counter): After reset, on the first T0STRT rising edge, TIMER 0 starts counting and at the next rising edge it stops. In this manner the period of an external signal may be measured.

The same above mentioned modes, can be used to reset the Timer0 by using the TRES pin signal.

The polarity of the T0SRTR Start/Stop signal can be changed by setting the STRPOL and RESPOL bits in the INT\_POL Configuration Register (01h bit 3 and 4). When these bits are set, the PWM/Timer 0 is Started/Set on the low level or in the falling edge of the signal applied in the pins.

The Timer output signal, TxOUT, is a signal with a frequency equal to the one of the 16 bit-Prescaler output signal, PRESCOUTx, divided by a 16-bit counter set by writing the Output Register couple PWMx\_COUNT\_OUT\_x.

**Note:** the contents of these registers upgrades the Timer counter after it stops counting. Since the register couple is written in two steps this can cause side effects. In order to avoid this, the user should write the MSB before writing the LSB: actually, the 16-bit value is latched in parallel when the LSB is written. By writing only the LSB (and MSB equal to 0), the PWM/Timer is used as with an 8 bit counter.

**Warning:** in Timer Mode the Reload Register couple PWMx\_RELOAD\_x (see PWM mode) must be set to the higher value FFFFh (65535) otherwise it can affect the count duration.

There can be two types of TxOUT waveforms:

- type 1: TxOUT waveform equal to a square wave with a 50% duty-cycle
- type 2: TxOUT waveform equal to a pulse signal with the pulse duration equal to the Prescaler output signal.

Figure 12.3 TxOUT Signal Types

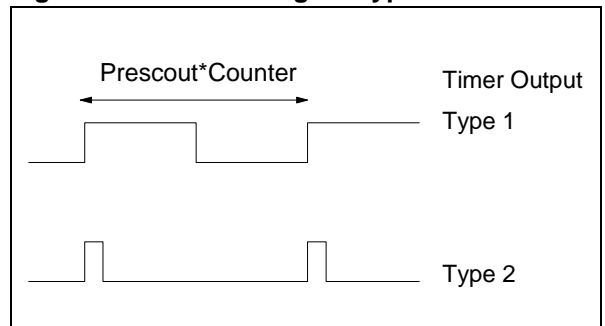
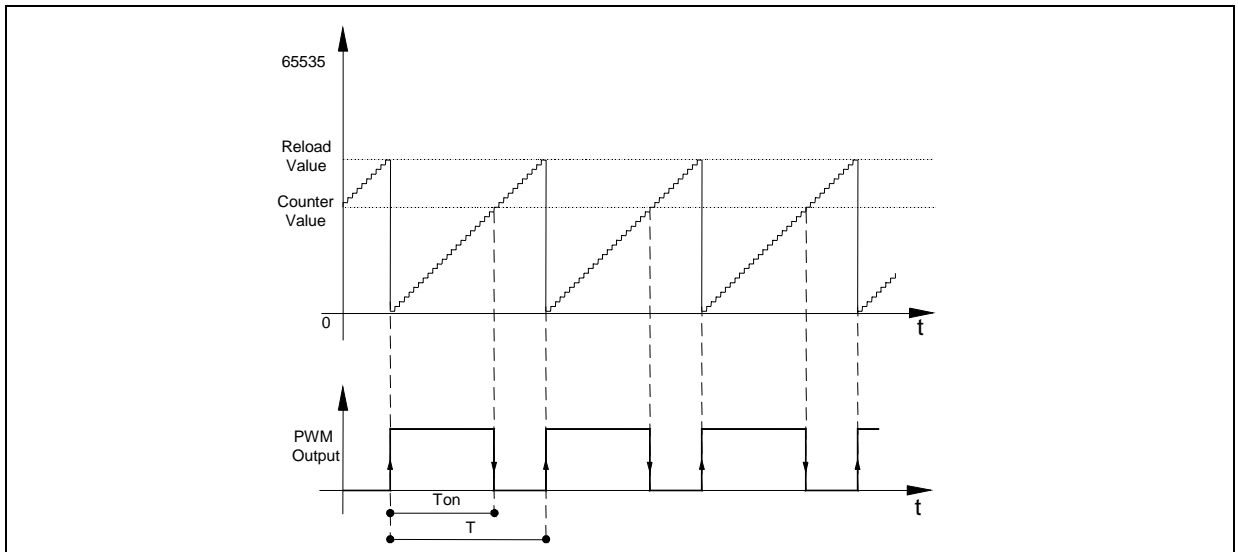




Figure 12.4 PWM Mode with Reload



### 12.3 PWM Mode

The PWM working mode for each timer is obtained by setting the TxMOD bit of the Configuration Register PWMx\_CR1.

The TxOUT signal in PWM Mode consists of a signal with a fixed period, whose duty cycle can be modified by the user.

The TxOUT period is fixed by setting the 16-bit Prescaler bits (TxPRESC) in the PWMx\_CR2 and the 16-bit Reload value by writing the relative Output Registers couple PWMx\_RELOAD\_x. The 16-bit Prescaler divides the master clock CLKM by powers of two, determining the maximum length period.

Reload determines the maximum value that the counter can count before starting a new period. The use of the two 16-bit values allows the TxOUT period to be set with more precision when needed. By setting the Reload value the counting resolution decreases. In order to obtain the maximum resolution, Reload value should be set to 0FFFFh and the period corresponds to the one established by the Prescaler value.

The value set in the 16-bit counter by writing the Counter Output Registers couple, determines the duty-cycle: when count reaches the Counter value the TxOUT signal changes from high to low level.

The period of the PWM signal is obtained by using the following formula:

$$T = \text{PWMx\_RELOAD} * 2^{\text{TxPRESC}} * \text{TMRCLKx}$$

where TxPRES equals the value set in the TxPRESC bits of the PWMx\_CR2 Configuration Register and TMRCLKx is the period of the Timer clock that drives the Prescaler.

The duty cycle of the PWM signal is obtained by the following formula:

$$d_{\text{cycle}} = \frac{T_{\text{on}}}{T} = \frac{\text{PWMx\_COUNT}}{\text{PWMx\_RELOAD}}$$

**Note:** the PWM\_x\_COUNT value must be lower than or equal to the PWM\_X\_RELOAD value. When it is equal, the TxOUT signal is always at high level. If the Output Register PWM\_x\_COUNT is 0, TxOUT signal is always at a low level.

By using a 24 MHz clock a PWM frequency that is close to 100 KHz can be obtained.

The TIMER0 clock CLKM can also be supplied with an external signal, applied on the TCLK pin, which must have a frequency that is at least two times smaller than the internal master clock.

**Note:** he Timers have to complete the previous counting phase before using a new value of the Counter. If the Counter value is changed during counting, the new values of the timer Counter are only used at the end of the previous counting phase. The Counter buffer is written in two steps (one byte per time) and is latched only after the LSB is written. In order to avoid side effects, the user should write the MSB before writing the LSB. By only writing the LSB, the PWM/Timer is used as with a 8 bit counter. The same mechanism is applied to the two bytes of Reload but, differently of the Counter it is set immediately. Nevertheless, it is recommended that the Reload value be written when the Timer is stopped in order to avoid incongruence with the Counter value. The same recommendation is made when reading the two bytes of the counter: It is performed in two steps, so if the timer is running, the carry of the LSB to the MSB can cause the wrong 16-bit value reading. A Reload value greater than 1 must always be used.

When the Timers are in Reset status, or when the device is reset, the TxOUT pins goes in threestate. If these outputs are used to drive external devices, it is recommended that the related pins be left in the default configuration (Input threestate) or change them in this configuration.

In PWM mode the PWM/Timers can only be Set or Reset: Start/Stop signals do not affect the Timers. TxRES resets the content of the 16-bit counter to zero. It is generated by writing 0 in the corresponding TxRES bit of the PWMx\_CR1 Configuration Register and/or it can be driven by the TRES pin if it is configured (only Timer0).

**12.3.1 Simultaneous Start.** The PWM/Timers can be started simultaneously when working in PWM mode. The T0SYNC and T1SYNC bits in PWM0\_CR3 Configuration Registers mask the reset of each timer; after enabling each single PWM/Timer. They are started by putting off the mask with a single writing in the PWM0\_CR3 Register.

Simultaneous start is also possible in Timer mode. The timers start counting simultaneously, but the output pulses are generated according to the modality configured (square or pulse mode).

**12.4 Timer Interrupts**

The PWM/Timer can be programmed to generate an Interrupt Request, both on the falling and the rising of the TxOUT signal and when there's a STOP signal (external or internal).

By using the TxIES, TxIER and TxIEF bits of the Configuration Registers PWMx\_CR1, the interrupt sources can be switched on/off. All the interrupt sources may be activated at the same time: sources can be distinguished by reading the PWMx\_STATUS Input Register.

The interrupt on the falling edge corresponds to half of a counting period in Timer mode when the waveform is set to Square Wave and to the end of the Ton phase in PWM mode.

**Note:** when the PWM Counter is set to 0 or 65535, the interrupt occurs at the end of each control period.

In order to be active, the PWM/Timers interrupts must be enabled by writing the Interrupt Mask Register (INT\_MASK) in the Configuration Register Space, bits MSKT0 And MSKT1.

**12.5 PWM/Timer 0 Register Description**

The following registers are related to the use of the PWM/Timer 0.

**12.5.1 PWM/Timer 0 Configuration Registers.**

**PWM/Timer 0 Control Register 1 (PWM0\_CR1)**

Configuration Register 9 (09h) Read/Write  
Reset Value: 0000 0000 (00h)

7							0
T0MOD	T0IES	T0IEF	T0IER	STRMOD	T0STRT	RESMOD	TORES

Bit 7: **T0MOD** PWM/Timer 0 Mode  
0: Timer Mode  
1: PWM Mode

Bit 6: **T0IES** Interrupt on Stop signal Enable  
0: interrupt disabled  
1: interrupt enabled

Bit 5: **T0IEF** Interrupt on T0OUT falling Enable  
0: interrupt disabled  
1: interrupt enabled

Bit 4: **T0IER** Interrupt on T0OUT rising Enable  
0: interrupt disabled  
1: interrupt enabled

Bit 3: **STRMOD** Start signal mode  
0: start on level  
1: start on edge

Bit 2: **T0STRT** PWM/Timer 0 Start bit  
0: Timer 0 stopped  
1: Timer 0 started

Bit 1: **RESMOD** Reset signal mode  
0: reset on level  
1: reset on edge

Bit 0: **TORES** PWM/Timer 0 Reset bit  
0: PWM/Timer 0 reset  
1: PWM/Timer 0 set



**PWM/Timer 0 Control Register 2 (PWM0\_CR2)**

Configuration Register 10 (0Ah) Read/Write  
Reset Value: 0000 0000 (00h)

7	4	0
-	-	T0WAV
		TOPRESC

Bit 7-6: Not Used

Bit 5: **T0WAV** T0OUT Waveform

- 0: pulse (type2)
- 1: square (type1)

Bit 4-0: **TOPRESC** PWM/Timer 0 Prescaler

The PWM/Timer 0 clock is divided by a factor equal to  $2^{TOPRESC}$ . **The maximum value allowed for TOPRESC is 10000 (010h).**

**PWM/Timer 0 Control Register 3 (PWM0\_CR3)**

Configuration Register 11 (0Bh) Read/Write  
Reset Value: 0000 0000 (00h)

7	4	2	0
T1SYNC	-	T0SYNC	T0CKS
		STRSRC	RESSRC

Bit 7: **T0SYNC** PWM/Timer 0 Set/Reset mask

- 0: Set/Reset activated
- 1: Set/Reset masked

Bit 6: not used

Bit 5: **T1SYNC** PWM/Timer 1 Set/Reset mask

- 0: Set/Reset activated
- 1: Set/Reset masked

Bit 4: **T0CKS** PWM/Timer 0 Clock Source

- 0: Internal clock
- 1: External Clock from TCLK

Bit 3-2: **STRSRC** PWM/Timer 0 Start signal source

- 00: Internal from T0STRT bit
- 01: External from TSTRT pin
- 10: Both internal and external

Bit 1-0: **RESSRC** PWM/Timer 0 Reset source

- 00: Internal from T0RES bit
- 01: External from TRES pin
- 10: Both internal and external

**Interrupt Polarity Register (INT\_POL)**

Configuration Register 1 (01h) Read/Write  
Reset Value: 0000 0000 (00h)

7	0
-	-
-	-
RESPOL	STRPOL
POLPB	POLPA
POLNMI	

Bit 7-5: Not Used

Bit 4: **RESPOL** Reset signal polarity

- 0: Reset on low level/rising edge
- 1: Reset on high level/falling edge

Bit 3: **STRPOL** Start signal polarity

- 0: Start on high level/rising edge
- 1: Start on low level/falling edge

Bit 2-0: See Interrupt Registers Description

**12.5.2 PWM/Timer 0 Input Registers.**

**PWM/Timer 0 Counter High Input Register (PWM0\_COUNT\_IN\_H)**

Input Register 21 (015h) Read only  
Reset Value: 0000 0000 (00h)

7	0
T0CI15	T0CI14
T0CI13	T0CI12
T0CI11	T0CI10
T0CI9	T0CI8

Bit 7-0: **T0CI15-8** PWM/Timer 0 Counter MSB

In this register the current value of the Timer 0 Counter MSB can be read.

**PWM/Timer 0 Counter Low Input Register (PWM0\_COUNT\_IN\_L)**

Input Register 22 (016h) Read only  
Reset Value: 0000 0000 (00h)

7								0
T0CI7	T0CI6	T0CI5	T0CI4	T0CI3	T0CI2	T0CI1	T0CI0	

Bit 7-0: **T0CI7-0** PWM/Timer 0 Counter MSB

In this register the current value of the Timer 0 Counter LSB can be read.

**PWM/Timer 0 Status Register (PWM0\_STATUS)**

Input Register 23 (017h) Read only  
Reset Value: 0000 0000 (00h)

7							0
-	-	-	-	T0OVFL	T0OUT	TORST	T0SST

Bit 7-4: Not Used

Bit 3: **T0OVFL** PWM/Timer 0 counter overflow flag  
0: no overflow occurred since last reset  
1: overflow occurred

Bit 2: **T0OUT** T0OUT pin value  
0: T0OUT pin is at logical level 0  
1: T0OUT pin is at logical level 1

Bit 1: **TORST** Reset Status  
0: PWM/Timer 0 is reset  
1: PWM/Timer 0 is set

Bit 0: **T0SST** Start Status  
0: PWM/Timer 0 is stopped  
1: PWM/Timer 0 is running

**PWM/Timer 0 Capture High Input Register (PWM0\_CAPTURE\_H)**

Input Register 24 (018h) Read only  
Reset Value: 0000 0000 (00h)

7								0
T0CP15	T0CP14	T0CP13	T0CP12	T0CP11	T0CP10	T0CP9	T0CP8	

Bit 7-0: **T0CP15-8** PWM/Timer 0 Capture MSB

In this register the counter value after the last stop can be read.

**PWM/Timer 0 Capture Low Input Register (PWM0\_CAPTURE\_L)**

Input Register 25 (019h) Read only  
Reset Value: 0000 0000 (00h)

7								0
T0CP7	T0CP6	T0CP5	T0CP4	T0CP3	T0CP2	T0CP1	T0CP0	

Bit 7-0: **T0CP7-0** PWM/Timer 0 Capture LSB

In this register the counter value after the last stop can be read.

**12.5.3 PWM/Timer 0 Output Registers.**

**PWM/Timer 0 Counter High Output Register (PWM0\_COUNT\_OUT\_H)**

Output Register 7 (07h) Write only  
Reset Value: 0000 0000 (00h)

7								0
T0CO15	T0CO14	T0CO13	T0CO12	T0CO11	T0CO10	T0CO9	T0CO8	

Bit 7-0: **T0CO15-8** PWM/Timer 0 Counter MSB  
This register is used to write the Timer 0 Counter value (MSB).

**Note:** this register is latched after writing the LSB part (PWM\_COUNT\_OUT\_L: see below). For this reason this register must be written before the LSB.

**PWM/Timer 0 Counter Low Output Register (PWM0\_COUNT\_OUT\_L)**

Output Register 8 (08h) Write only  
Reset Value: 0000 0000 (00h)

7								0
T0CO7	T0CO6	T0CO5	T0CO4	T0CO3	T0CO2	T0CO1	T0CO0	

Bit 7-0: **T0CO7-0** PWM/Timer 0 Counter MSB

This register is used to write the Timer 0 Counter value (LSB).

**Note:** writing this register, the *PWM0\_COUNT\_OUT\_x* couple is latched in the internal registers of the peripherals. For this reason, this register should be written after the MSB one.

**PWM/Timer 0 Reload High Output Register (PWM0\_RELOAD\_H)**

Output Register 9 (09h) Write only

Reset Value: 1111 1111 (0FFh)

7								0
T0REL15	T0REL14	T0REL13	T0REL12	T0REL11	T0REL10	T0REL9	T0REL8	

Bit 7-0: **T0REL15-8** PWM/Timer 0 Reload MSB

This register is used to write the Timer 0 Reload value (MSB).

**Note:** this register is latched after writing the LSB part (*PWM0\_RELOAD\_L*: see below). For this reason, this register must be written before the LSB.

**PWM/Timer 0 Reload Low Output Register (PWM0\_RELOAD\_L)**

Output Register 10 (0Ah) Write only

Reset Value: 1111 1111 (0FFh)

7								0
T0REL7	T0REL6	T0REL5	T0REL4	T0REL3	T0REL2	T0REL1	T0REL0	

Bit 7-0: **T0REL7-0** PWM/Timer 0 Reload LSB

This register is used to write the Timer 0 Reload value (LSB).

**Note:** by writing this register, the *PWM0\_RELOAD\_x* couple is latched in the internal registers of the peripherals. For this reason this register should be written after the MSB one.

**12.6 PWM/Timer 1 Register Description**

The following registers are related to the use of the PWM/Timer 1.

**12.6.1 PWM/Timer 1 Configuration Registers.**

**PWM/Timer 1 Control Register 1 (PWM1\_CR1)**

Configuration Register 12 (0Ch) Read/Write

Reset Value: 0000 0000 (00h)

7							0
T1MOD	T1IES	T1IEF	T1IER	-	T1STRT	-	T1RES

Bit 7: **T1MOD** PWM/Timer 1 Mode

0: Timer Mode

1: PWM Mode

Bit 6: **T1IES** Interrupt on Stop signal Enable

0: interrupt disabled

1: interrupt enabled

Bit 5: **T1IEF** Interrupt on T1OUT falling Enable

0: interrupt disabled

1: interrupt enabled

Bit 4: **T1IER** Interrupt on T1OUT rising Enable

0: interrupt disabled

1: interrupt enabled

Bit 3: not used

Bit 2: **T1STRT** PWM/Timer 1 Start bit

0: Timer 0 stopped

1: Timer 0 started

Bit 1: not used

Bit 0: **T1RES** PWM/Timer 1 Reset bit

0: PWM/Timer 0 reset

1: PWM/Timer 0 set

**PWM/Timer 1 Control Register 2 (PWM1\_CR2)**

Configuration Register 13 (0Dh) Read/Write  
Reset Value: 0000 0000 (00h)

7	5	0
-	-	T1WAV
		T1PRESC

Bit 7-6: Not Used

Bit 5: **T1WAV** T1OUT Waveform

- 0: pulse (type2)
- 1: square (type1)

Bit 4-0: **T1PRESC** PWM/Timer 1 Prescaler

The PWM/Timer 1 clock is divided by a factor equal to  $2^{T1PRESC}$ . **The maximum value allowed for T1PRESC is 1000 (010h).**

**12.6.2 PWM/Timer 1 Input Registers.**

**PWM/Timer 1 Counter High Input Register (PWM1\_COUNT\_IN\_H)**

Input Register 26 (01Ah) Read only  
Reset Value: 0000 0000 (00h)

7	0
T1CI15	T1CI14
T1CI13	T1CI12
T1CI11	T1CI10
T1CI9	T1CI8

Bit 7-0: **T1CI15-8** PWM/Timer 1 Counter MSB

In this register the current value of the Timer 1 Counter MSB can be read.

**PWM/Timer 1 Counter Low Input Register (PWM1\_COUNT\_IN\_L)**

Input Register 27 (01Bh) Read only  
Reset Value: 0000 0000 (00h)

7	0
T1CI7	T1CI6
T1CI5	T1CI4
T1CI3	T1CI2
T1CI1	T1CI0

Bit 7-0: **T1CI7-0** PWM/Timer 1 Counter LSB

In this register the current value of the Timer 1 Counter LSB can be read.

**PWM/Timer 1 Status Register (PWM1\_STATUS)**

Input Register 28 (01Ch) Read only  
Reset Value: 0000 0000 (00h)

7	0
-	-
-	-
-	-
T1OVFL	T1OUT
T1RST	T1SST

Bit 7-4: Not Used

Bit 3: **T1OVFL** PWM/Timer 1 counter overflow flag

- 0: no overflow occurred since last reset
- 1: overflow occurred

Bit 2: **T1OUT** T1OUT pin value

- 0: T1OUT pin is at logical level 0
- 1: T1OUT pin is at logical level 1

Bit 2: **T1RST** Reset Status

- 0: PWM/Timer 1 is reset
- 1: PWM/Timer 1 is set

Bit 2: **T1SST** Start Status

- 0: PWM/Timer 1 is stopped
- 1: PWM/Timer 1 is running

**PWM/Timer 1 Capture High Input Register (PWM1\_CAPTURE\_H)**

Input Register 29 (01Dh) Read only  
Reset Value: 0000 0000 (00h)

7	0
T1CP15	T1CP14
T1CP13	T1CP12
T1CP11	T1CP10
T1CP9	T1CP8

Bit 7-0: **T1CP15-8** PWM/Timer 1 Capture MSB

In this register the counter value after the last stop can be read.

**PWM/Timer 1 Capture Low Input Register (PWM1\_CAPTURE\_L)**

Input Register 30 (01Eh) Read only  
Reset Value: 0000 0000 (00h)

7								0
T1CP7	T1CP6	T1CP5	T1CP4	T1CP3	T1CP2	T1CP1	T1CP0	

Bit 7-0: **T1CP7-0** PWM/Timer 1 Capture LSB

In this register the counter value after the last stop can be read.

**12.6.3 PWM/Timer 1 Output Registers.****PWM/Timer 1 Counter High Output Register (PWM1\_COUNT\_OUT\_H)**

Output Register 11 (0Bh) Write only  
Reset Value: 0000 0000 (00h)

7								0
T1CO15	T1CO14	T1CO13	T1CO12	T1CO11	T1CO10	T1CO9	T1CO8	

Bit 7-0: **T1CO15-8** PWM/Timer 1 Counter MSB

This register is used to write the Timer 1 Counter value (MSB).

**Note:** this register is latched after writing the LSB part (PWM1\_COUNT\_OUT\_L: see below). For this reason, this register must be written before the LSB.

**PWM/Timer 1 Counter Low Output Register (PWM1\_COUNT\_OUT\_L)**

Output Register 12 (0Ch) Write only  
Reset Value: 0000 0000 (00h)

7								0
T1CO7	T1CO6	T1CO5	T1CO4	T1CO3	T1CO2	T1CO1	T1CO0	

Bit 7-0: **T1CO7-0** PWM/Timer 0 Counter MSB

This register is used to write the Timer 1 Counter value (LSB).

**Note:** by writing this register, the PWM1\_COUNT\_OUT\_x couple is latched in the internal registers of the peripherals. For this reason this register should be written after the MSB one.

**PWM/Timer 1 Reload High Output Register (PWM1\_RELOAD\_H)**

Output Register 13 (0Dh) Write only  
Reset Value: 1111 1111 (0FFh)

7								0
T1REL15	T1REL14	T1REL13	T1REL12	T1REL11	T1REL10	T1REL9	T1REL8	

Bit 7-0: **T1REL15-8** PWM/Timer 0 Reload MSB

This register is used to write the Timer 1 Reload value (MSB).

**Note:** this register is latched after writing the LSB part (PWM1\_RELOAD\_L: see below). For this reason, this register must be written before the LSB.

**PWM/Timer 1 Reload Low Output Register (PWM0\_RELOAD\_L)**

Output Register 14 (0Eh) Write only  
Reset Value: 1111 1111 (0FFh)

7								0
T1REL7	T1REL6	T1REL5	T1REL4	T1REL3	T1REL2	T1REL1	T01RELO	

Bit 7-0: **T1REL7-0** PWM/Timer 1 Reload LSB

This register is used to write the Timer 1 Reload value (LSB).

**Note:** by writing this register, the PWM1\_RELOAD\_x couple is latched in the internal registers of the peripherals. For this reason, this register should be written after the MSB one.

### 13 SERIAL COMMUNICATION INTERFACE

The Serial Communication Interface (SCI) integrated into ST52F510/F513/F514 provides a general purpose shift register peripheral, several widely distributed devices to be linked, through their SCI subsystem. SCI gives a serial interface providing communication with the speed from less than 300 up to over 115200 baud, and a flexible character format.

SCI is a full-duplex UART-type asynchronous system with standard Non Return to Zero (NRZ) format for the transmitted/received bit. The length of the transmitted word is 10/11 bits (1 start bit, 8/9 data bits, 1 stop bit).

SCI is composed of three modules: Receiver, Transmitter and Baud-Rate Generator.

#### 13.1 SCI Receiver block

The SCI Receiver block manages the synchronization of the serial data stream and stores the data characters. The SCI Receiver is mainly composed of two sub-systems: Recovery Buffer Block and SCDR\_RX Block.

SCI receives data deriving from the RX pin and drives the Recovery Buffer Block, which is a high-speed shift register operating at a clock frequency (CLOCK\_RX) 16 times higher than the fixed baud rate (CLOCK\_TX). This sampling rate, higher than the Baud Rate clock, detects the START condition, Noise error and Frame error.

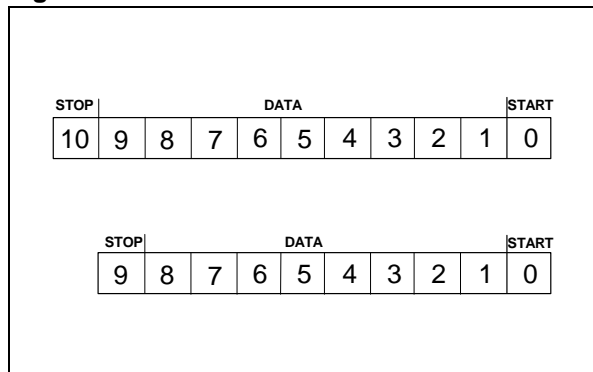
When the SCI Receiver is in IDLE status, it is waiting for the START condition, which is obtained with a logic level of 0, consecutive to a logic level 1. This condition is detected if, with the fixed

sampling time, a logic level 0 is sampled after three logic levels of 1.

The recognition of the START bit forces the SCI Receiver Block to start a data acquisition sequence.

The data acquisition sequence is configured by the apposite Configuration Register, allowing the following data frame formats (see Figure 13.1):

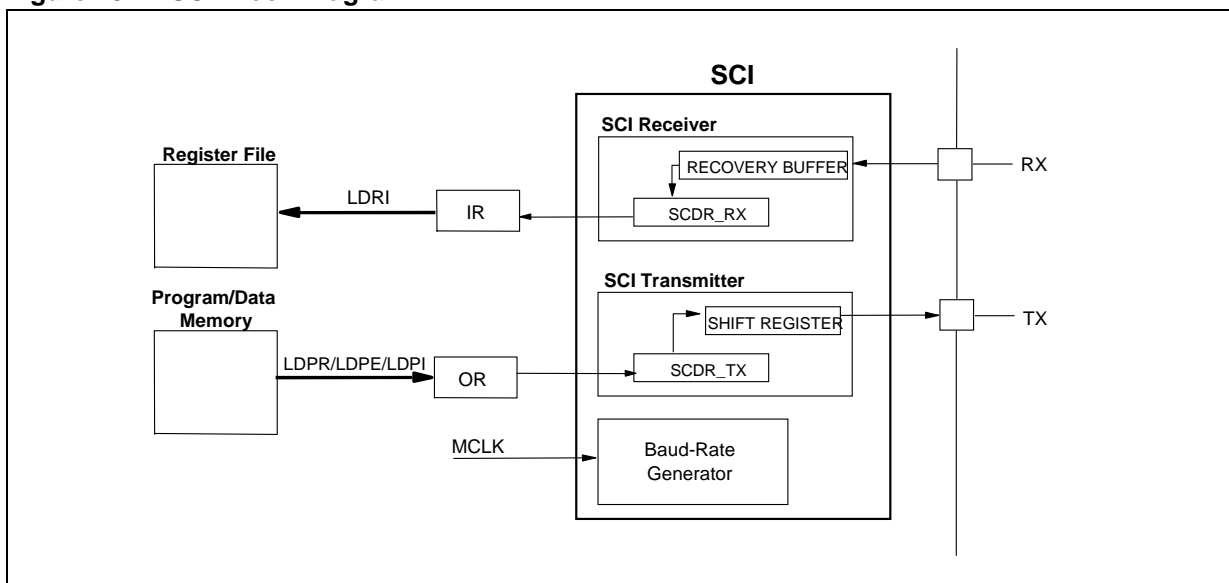
Figure 13.1 SCI transmitted word structures



- 8 bit length, 1 stop bit, no parity bit
- 8 bit length, 2 stop bit, no parity bit
- 8 bit length, 1 stop bit, with parity bit
- 9 bit length, 1 stop bit, no parity bit

The parity bit (if used) can be configured for even or odd parity check. If the 9-bit length format is configured, this bit is used in transmission for the ninth bit (see below). The ninth bit received can be read in the R8 bit of the SCI Status Register, address 37 (035h) bit 2 (see Figure 13.3).

Figure 13.2 SCI Block Diagram





Recognition of a STOP condition transfers data received from the Recovery Buffer to the SCDR\_RX buffer, adding the eventual ninth data bit. After this operation, RXF flag (bit 5) of SCI Status Input Register is set to logic level 1. The Control Unit reads data from the SCDR\_RX buffer (in read-only mode) by reading the SCI\_IN Input Register (address 36 024h) with the LDRI instruction and provides a reset at logic level 0 to the RXF flag.

If data of the Recovery Buffer is ready to be transferred into the SCDR\_RX buffer, but the previous one has not been read by the Core, an OVERRUN Error takes place: the SCI Status Register flag OVERR (bit 4) indicates the error condition. In this case, information that is stored in the SCDR\_RX buffer is not altered, but the one that has caused the OVERRUN error can be overwritten by new data deriving from the serial data line.

### 13.1.1 Recovery Buffer Block .

This block is structured as a synchronized finite state machine on the CLOCK\_RX signal.

When the Recovery Buffer Block is in IDLE state it waits for the reception of the correct 1 and 0 sequence representing START.

Recognition takes place by sampling the input RX at CLOCK\_RX frequency, which has a frequency that is 16 times higher than CLOCK\_TX. For this reason, while the external transmitter sends a single bit, the Recovery Buffer Block samples 16 states (from SAMPLE1 to SAMPLE16).

Analysis of the RX input signal is carried out by checking three samples for each bit received.

If these three samples are not equal, then the noise error flag, NSERR (bit 7), of SCI Status Register is set to 1 and the data received value will be the one assumed by the majority of the samples.

The procedure described above, allows SCI not to become IDLE, because of a limited noise due to an erroneous sampling, the transmission is recognized as correct and the noise flag error is set.

At the end of the cycle of the reception of a bit, the Recovery Buffer Block will repeat the same steps 9 times: one step for each bit received, plus one for the stop acquisition (10 times in case of 9-bit data, double stop or parity check).

At the end of data reception the Recovery Buffer Block will supply information about eventual frame errors by setting the 1 FRERR flag (bit 6) of the SCI Status Register to 1.

A frame error can occur if the parity check hasn't been successfully achieved or if the STOP bit has not been detected.

If the Recovery Buffer Block receives 10 consecutive bits at logic level 0, a Line Break condition occurs and the related Interrupt Request is sent.

### 13.1.2 SCDR\_RX Block.

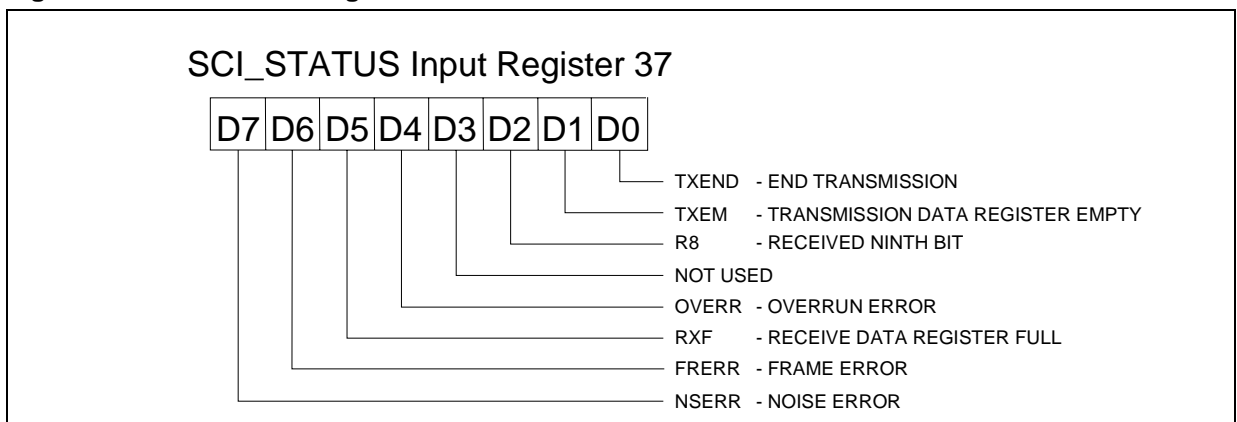
It is a finite state machine synchronized with the clock master signal, CKM.

The SCDR\_RX block waits for the signal of complete reception from the Recovery Buffer in order to load the word received. Moreover, the SCDR\_RX block loads the values of FRERR and NSERR flag bits of the Status Register, and sets the RXF flag to 1.

By using the LDRI instruction data is transferred to Register File and RXF flag is reset to 0, to indicate that the SCDR\_RX block is empty.

If new data arrives before the previous one has been transferred to Register File, the overrun error occurs and the OVERR flag of Status Register is set to 1.

Figure 13.3 SCI Status Register



### 13.2 SCI Transmitter Block

The SCI Transmitter Block consists of the following blocks: SCDR\_TX and SHIFT REGISTER, synchronized, respectively, with the clock master signal (CKM) and the CLOCK\_TX.

The whole block receives the settings for the following transmission modes through the Configuration Register:

- 8 bit length, 1 stop bit, no parity bit
- 8 bit length, 2 stop bit, no parity bit
- 8 bit length, 1 stop bit, with parity bit
- 9 bit length, 1 stop bit, no parity bit

In case of 9 bit frame transmission, the most significative bit arrives through the bit PAR/T8 (bit 2) of the SCI\_CR1 Configuration Register. In an 8-bit transmission, instead, this bit is used to configure the data format: in particular to choose the polarity control (even or odds) to implement the parity check (see above).

After a RESET, the SCDR\_TX block is in IDLE state until it receives an enabling signal by writing the TXSTRT bit of the SCI\_CR2 Configuration Register.

The data is loaded on the Peripheral Register SCI\_OUT (address 23 017h) by using the instruction LPPR, LDPI or LDPE. If the transmission is enabled, the data to be transmitted is transferred from the Output Register to SCDR\_TX block and the TXEM flag (bit 1) of the SCI Status Register is reset to 0 to indicate SCDR\_TX block is full.

If the core supplies new data, this could not be loaded in the SCDR\_TX block until the current data has not been unloaded on the Shift Register block. Meaning that only when TXEM is 1 data can be loaded in the SCDR\_TX Block.

When the SHIFT REGISTER Block loads the data to be transmitted on an internal buffer, the TXEND flag (bit 0) of the SCI Status Register is reset to 0 to indicate the beginning of a new transmission. At the end of transmission TXEND is set to 1, allowing new data coming from SCDR\_TX to be loaded in the SHIFT REGISTER.

It is important to underline that TXEND = 1 does not mean SCDR\_TX is ready to receive a new data. For this reason, it is better to utilize the TXEM signal to synchronize the load instruction to the SCI TRANSMITTER block

If the TXSTRT bit is reset, the transmission is stopped, but the SCI Transmitter block completes the transmission in progress before resetting.

### 13.3 Baud Rate Generator Block

The Baud Rate Generator Block performs the division of the clock master signal (CKM) in a set of synchronism frequencies for the serial bit reception/transmission on the external line.

Reception frequency (CLOCK\_RX) is 16 times higher than the transmission frequency (CLOCK\_TX).

To adapt the Baud Rate Generator to the clock master frequency supplied by the user, a 12-bit Prescaler must be programmed by loading the Configuration Registers SCI\_CR2 (PRESC\_H bit 11:8 of the 12 bit prescaler) and SCI\_CR3 (PRESC\_L bit 7:0 of the 12 bit prescaler). The prescaler allows the programming of all standard Baud Rates by using the most common clock master sources.

The Prescaler value can be obtained by the following formula:

$$PRESC = round\left(\frac{CKM}{16 \times BAUD}\right)$$

Where CKM is the clock master frequency (expressed in Hz) and BAUD is the desired Baud Rate (expressed in bit/second). The obtained value is rounded to the nearest integer value. This rounding can cause an error in the obtained Baud Rate. This error must be lower than 3%. To verify that the PRESC value satisfies this constrain, the obtained Baud Rate must be computed by inverting the previous formula:

$$\overline{BAUD} = \frac{CKM}{16 \times PRESC}$$

then the following relation can be used to verify that the difference with the desired Baud Rate is lower than 3%:

$$\frac{|BAUD - \overline{BAUD}|}{BAUD} < 0.03$$

Table 13.1 shows the recommended Prescaler values for common clock master frequencies. To get more precision in Baud Rate, standard quartz frequencies for serial communication can be used. The corresponding Prescaler values for these frequencies are showed in the Table 13.2.

**Table 13.1 Recommended Prescaler values for common frequencies (Baud/MHz)**

	1	4	5	8	10	12	16	20	24
<b>1200</b>	52	208	260	417	521	625	833	1042	1250
<b>2400</b>	26	104	130	208	260	313	417	521	625
<b>4800</b>	13	52	65	104	130	156	208	260	313
<b>9600</b>	-	26	33	52	65	78	104	130	156
<b>19200</b>	-	13	16	26	33	39	52	65	78
<b>38400</b>	-	-	8	13	16	20	26	33	39
<b>57600</b>	-	-	-	-	11	13	17	22	26
<b>115200</b>	-	-	-	-	-	-	-	11	13

**Table 13.2 Recommended Prescaler values for serial communication quartz (Baud/MHz)**

	1.843	2.458	3.686	4.915	6.144	7.373	9.830	11.059	12.288	14.746	19.661	22.118
<b>1200</b>	96	128	192	256	320	384	512	576	640	768	1024	1152
<b>2400</b>	48	64	96	128	160	192	256	288	320	384	512	576
<b>4800</b>	24	32	48	64	80	96	128	144	160	192	256	288
<b>9600</b>	12	16	24	32	40	48	64	72	80	96	128	144
<b>19200</b>	6	8	12	16	20	24	32	36	40	48	64	72
<b>38400</b>	3	4	6	8	10	12	16	18	20	384	32	36
<b>57600</b>	2	-	4	-	-	8	-	12	13	16	21	24
<b>115200</b>	1	-	2	-	-	4	-	6	-	8	-	12

**13.4 SCI Register Description**

The following registers are related to the use of the SCI peripheral.

**13.4.1 SCI Configuration Registers.**

**SCI Control Register 1 (SCI\_CR1)**

Configuration Register 22 (016h) Read/Write  
Reset Value: 0000 0000 (00h)

7						2	0
RXFINT	OVRINT	BRKINT	TXEMINT	TXENINT	PAR/T8	FRM	

Bit 7: **RXFINT** SCDR\_RX buffer full interrupt mask  
0: interrupt disabled  
1: interrupt enabled

Bit 6: **OVRINT** Overrun interrupt mask  
0: interrupt disabled  
1: interrupt enabled

Bit 5: **BRKINT** Break interrupt mask  
0: interrupt disabled  
1: interrupt enabled

Bit 4: **TXEMINT** SCDR\_TX buffer empty interrupt  
0: interrupt disabled  
1: interrupt enabled

Bit 3: **TXENINT** TX end interrupt mask  
0: interrupt disabled  
1: interrupt enabled

Bit 2: **PAR/T8** Parity type selection or TX 9th bit  
0: parity odd if enabled, else TX 9th bit=0  
1: parity even if enabled, else TX 9th bit=1

Bit 1-0: **FRM** Frame type selection  
00: 8 bit, no parity, 1 stop bit  
01: 8 bit, no parity, 2 stop bit  
10: 8 bit, parity, 1 stop bit  
11: 9 bit, no parity, 1 stop bit

**Note:** the SCI interrupts are not enabled unless the bit 3 (MSKSCI) of the Configuration Register 0 (INT\_MASK) is enabled (set to 1).

**SCI Control Register 2 (SCI\_CR2)**

Configuration Register 23 (017h) Read/Write  
Reset Value: 0000 0000 (00h)

7			4			2	0
PRESC_H			-		RXSTRT	TXSTRT	

Bit 7-4: **PRESC\_H** Baud Rate prescaler (bit 11:8)  
These bits are the higher part of the prescaler (see SCI\_CR3 Configuration Register) which determinates the baud rate of the communication, according to Table 13.1 and Table 13.2, as explained in Paragraph 13.3.

Bit 3-2: not used

Bit 1: **RXSTRT** Reception enable  
0: RX disabled  
1: RX enabled

Bit 0: **TXSTRT** Transmission enable  
0: TX disabled  
1: TX enabled

**SCI Control Register 3 (SCI\_CR3)**

Configuration Register 43 (02Bh) Read/Write  
Reset Value: 0000 0000 (00h)

7							0
PRESC_L							

Bit 7-0: **PRESC\_L** Baud Rate prescaler (bit 7:0)  
These bits are the lower part of the prescaler (see SCI\_CR2 Configuration Register) which determinates the baud rate of the communication, according to Table 13.1 and Table 13.2, as explained in Paragraph 13.3.

### 13.4.2 SCI Input Registers.

#### SCI RX data Input Register (SCI\_IN)

Input Register 36 (024h) Read only

Reset Value: 0000 0000 (00h)

7							0
RX7	RX6	RX5	RX4	RX3	RX2	RX1	RX0

Bit 7-0: **RX7-0** RX Data

In this register the last received serial data can be read.

#### SCI Status Register (SCI\_STATUS)

Input Register 37 (025h) Read only

Reset Value: 0000 0011 (03h)

7							0
NSERR	FRERR	RXF	OVERR	-	R8	TXEM	TXEND

Bit 7: **NSERR** Noise error

0: noise error not occurred

1: noise error occurred

Bit 6: **FRERR** Frame error

0: frame error not occurred

1: frame error occurred

Bit 5: **RXF** RX data register full

0: RX data register already read

1: RX data register full but not read yet

Bit 4: **OVERR** Overrun error

0: overrun error not occurred

1: overrun error occurred

Bit 3: not used

Bit 2: **R8** Received 9th bit

0: RX 9th bit=0

1: RX 9th bit=1

Bit 1: **TXEM** TX data register empty

0: TX data register full

1: TX data register empty

Bit 0: **TXEND** TX end flag

0: data transferred to the shift register

1: data transmission completed

### 13.4.3 SCI Output Register.

#### SCI TX data Output Register (SCI\_OUT)

Input Register 23 (017h) Write only

Reset Value: 0000 0000 (00h)

7							0
TX7	TX6	TX5	TX4	TX3	TX2	TX1	TX0

Bit 7-0: **TX7-0** TX Data

In this register the serial data to be transmitted can be written.

## 14 I<sup>2</sup>C BUS INTERFACE (I<sup>2</sup>C)

### 14.1 Introduction

The I<sup>2</sup>C Bus Interface serves as an interface between the microcontroller and the serial I<sup>2</sup>C bus, providing both multimaster and slave functions and controls all I<sup>2</sup>C bus-specific sequencing, protocol, arbitration and timing. The I<sup>2</sup>C Bus Interface supports fast I<sup>2</sup>C mode (400kHz).

### 14.2 Main Features

- Parallel-bus/I<sup>2</sup>C protocol converter
- Multi-master capability
- 7-bit/10-bit Addressing
- Transmitter/Receiver flag
- End-of-byte transmission flag
- Transfer problem detection

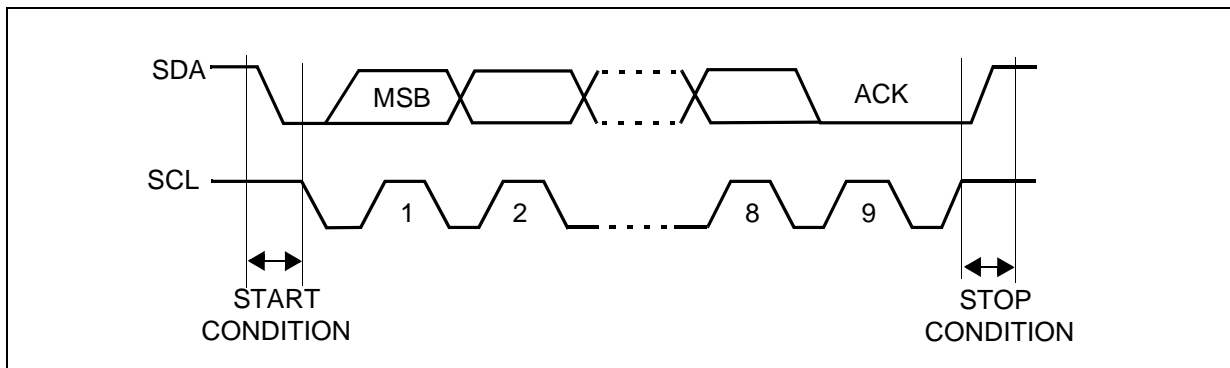
I<sup>2</sup>C Master Features:

- Clock generation
- I<sup>2</sup>C bus busy flag
- Arbitration Lost Flag
- End of byte transmission flag
- Transmitter/Receiver Flag
- Start bit detection flag
- Start and Stop generation

I<sup>2</sup>C Slave Features:

- Stop bit detection
- I<sup>2</sup>C bus busy flag
- Detection of misplaced start or stop condition
- Programmable I<sup>2</sup>C Address detection
- Transfer problem detection
- End-of-byte transmission flag
- Transmitter/Receiver flag

Figure 14.1 I<sup>2</sup>C BUS Protocol



### 14.3 General Description

In addition to receiving and transmitting data, this interface converts it from serial to parallel format and vice versa, using either an interrupt or polled handshake. The interrupts are enabled or disabled via software. The interface is connected to the I<sup>2</sup>C bus by a data pin (SDA) and by a clock pin (SCL). The interface can be connected both with a standard I<sup>2</sup>C bus and a Fast I<sup>2</sup>C bus. This selection is made via software.

#### 14.3.1 Mode Selection.

The interface can operate in the following four modes:

- Slave transmitter/receiver
- Master transmitter/receiver

By default, it operates in slave mode.

The interface automatically switches from slave to master after it generates a START condition and from master to slave in case of arbitration loss or a STOP generation, providing Multi-Master capability.

#### 14.3.2 Communication Flow.

In Master mode, Communication Flow initiates data transfer and generates the clock signal. A serial data transfer always begins with a start condition and ends with a stop condition. Both start and stop conditions are generated in master mode by software.

In Slave mode the interface is capable of recognizing its own address (7 or 10-bit) and the General Call address. The General Call address detection may be enabled or disabled by software. Data and addresses are transferred as 8-bit bytes, (MSB first). The first byte(s) follow the start condition is the address (one in 7-bit mode, two in 10-bit mode), which is always transmitted in Master mode. A 9th clock pulse follows the 8 clock cycles of a byte transfer, during which the receiver must send an acknowledge bit to the transmitter. Refer to Figure 14.1.

Acknowledge may be enabled and disabled via software.

The I<sup>2</sup>C interface address and/or general call address can be selected via software.

The speed of the I<sup>2</sup>C interface may be selected between Standard (0-100KHz) and Fast I<sup>2</sup>C (100-400KHz).

### 14.3.3 SDA/SCL Line Control.

*Transmitter mode:* the interface holds the clock line low before transmission, in order to wait for the microcontroller to write the byte in the Data Register.

*Receiver mode:* the interface holds the clock line low after reception to wait for the microcontroller to read the byte in the Data Register.

SCL frequency is controlled by a programmable clock divider which depends on the I<sup>2</sup>C bus mode.

When the I<sup>2</sup>C cell is enabled, the SDA and SCL pins must be configured as floating open-drain I/O.

The value of the external pull-up resistance used depends on the application.

## 14.4 Functional Description

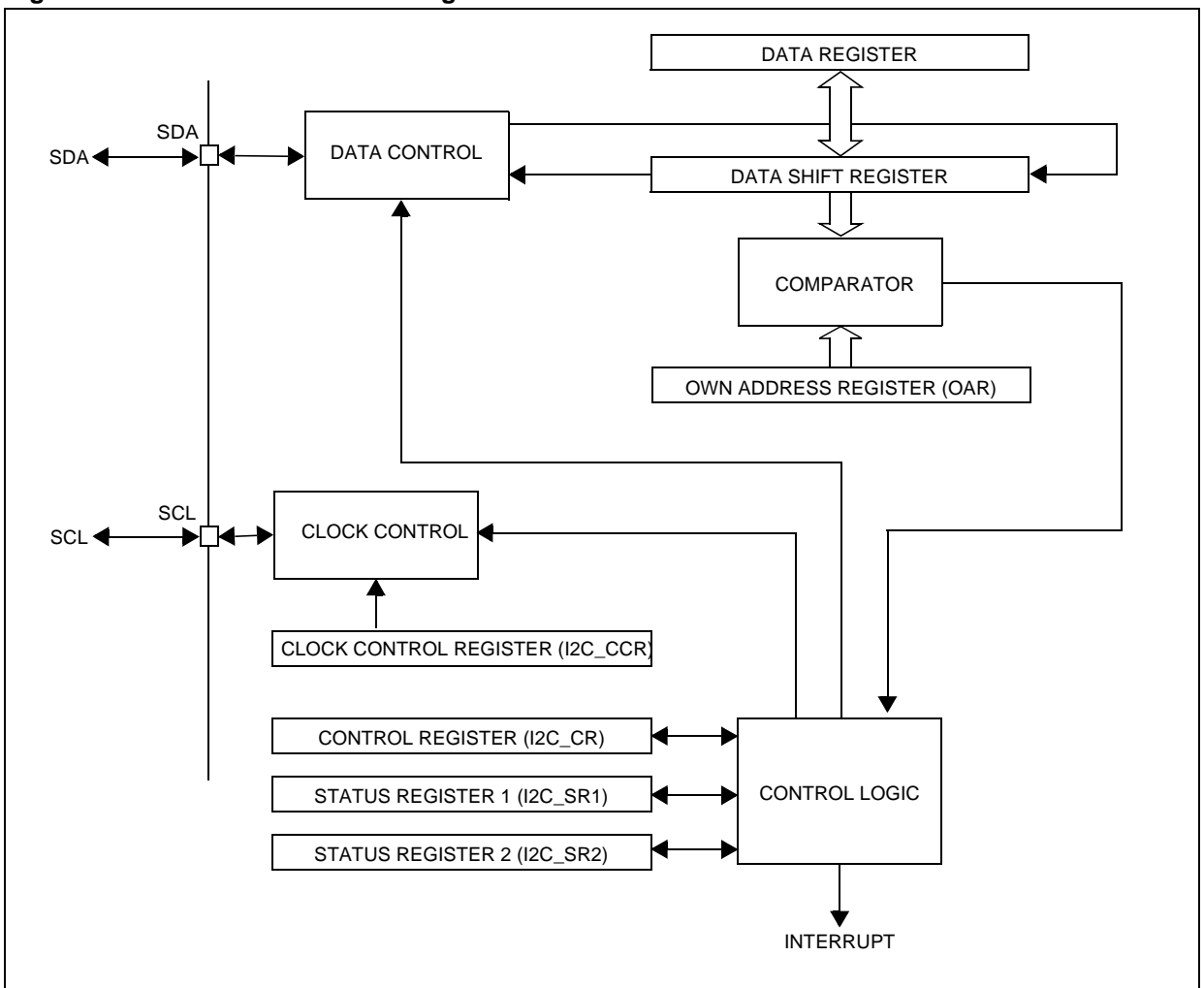
By default the I<sup>2</sup>C interface operates in Slave mode (M/SL bit is cleared) except when it initiates a transmit or receive sequence.

First, the interface frequency must be configured using the related bits of the Configuration Registers.

### 14.4.1 Slave Mode.

As soon as a start condition is detected, the address is received from the SDA line and sent to the shift register; then it is compared with the address of the interface or the General Call address (if selected by software).

Figure 14.2 I<sup>2</sup>C Interface Block Diagram



**Note:** In 10-bit addressing mode, the comparison includes the header sequence (11110xx0) and the two most significant bits of the address.

**Header matched** (10-bit mode only): the interface generates an acknowledgement pulse if the ACK bit is set.

**Address not matched:** the interface ignores it and waits for another Start condition.

**Address matched:** the interface generates in sequence:

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

Afterwards, the interface waits for the I2C\_SR1 register to be read, **holding the SCL line low** (see Figure 14.3 Transfer sequencing EV1).

Next, in 7-bit mode read the I2C\_IN register to determine from the least significant bit (Data Direction Bit) if the slave must enter Receiver or Transmitter mode.

In 10-bit mode, after receiving the address sequence the slave is always in receive mode. It will enter transmit mode on receiving a repeated Start condition followed by the header sequence with matching address bits and the least significant bit set (11110xx1).

### Slave Receiver

Following reception of the address and after the I2C\_SR1 register has been read, the slave receives bytes from the SDA line into the I2C\_IN register via the internal shift register. After each byte, the interface generates the following 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.

Afterwards, the interface waits for the I2C\_SR1 register to be read followed by a read of the I2C\_IN register, **holding the SCL line low** (see Figure 14.3 Transfer sequencing EV2).

### Slave Transmitter

Following the address reception and after the I2C\_SR1 register has been read, the slave sends bytes from the I2C\_OUT register to the SDA line via the internal shift register.

The slave waits for a read of the I2C\_SR1 register followed by a write in the I2C\_OUT register, **holding the SCL line low** (see Figure 14.3 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.

Afterwards, the interface waits for a read of the I2C\_SR2 register (see Figure 14.3 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.

**Note:** In both cases, the SCL line is not held low; however, SDA line can remain low due to possible «0» bits transmitted last. At this point, both lines must be released 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 current byte is transferred.

#### 14.4.2 Master Mode.

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

### Start condition

Setting the START bit while the BUSY bit is cleared causes the interface to switch to Master mode (M/SL bit set) and generates a Start condition.

Once the Start condition is sent:

- The EVF and SB bits are set by hardware with an interrupt if the ITE bit is set.

Afterwards, the master waits for a read of the I2C\_SR1 register followed by a write in the I2C\_OUT register with the Slave address, **holding the SCL line low** (see Figure 14.3 Transfer sequencing EV5).



### Slave address transmission

At this point, the slave address is sent to the SDA line via the internal shift register.

In 7-bit addressing mode, one address byte is sent.

In 10-bit addressing mode, sending the first byte including the header sequence causes the following event:

- The EVF bit is set by hardware with interrupt generation if the ITE bit is set.

Afterwards, the master waits for a read of the I2C\_SR1 register followed by a write in the I2C\_OUT register, **holding the SCL line low** (see Figure 14.3 Transfer sequencing EV9).

The second address byte is sent by the interface.

After completion of this transfer (and acknowledge from the slave if the ACK bit is set):

- The EVF bit is set by hardware with interrupt generation if the ITE bit is set.

Afterwards, the master waits for a read of the I2C\_SR1 register followed by a write in the I2C\_CR register (for example set PE bit), **holding the SCL line low** (see Figure 14.3 Transfer sequencing EV6).

Next, the master must enter Receiver or Transmitter mode.

**Note:** In 10-bit addressing mode, in order to switch the master to Receiver mode, 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 I2C\_SR1 and I2C\_CR registers have been accessed, the master receives bytes from the SDA line into the I2C\_IN 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.

Afterwards, the interface waits for a read of the I2C\_SR1 register followed by a read of the I2C\_IN register, **holding the SCL line low** (see Figure 14.3 Transfer sequencing EV7).

In order to close the communication: before reading the last byte from the I2C\_IN register, set the STOP bit to generate the Stop condition. The interface automatically goes back to slave mode (M/SL bit cleared).

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

### Master Transmitter

Following the address transmission and after the I2C\_SR1 register has been read, the master sends bytes from the I2C\_OUT register to the SDA line via the internal shift register.

The master waits for a read of the I2C\_SR1 register followed by a write in the I2C\_OUT register, **holding the SCL line low** (see Figure 14.3 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.

In order to close the communication: after writing the last byte to the I2C\_OUT register, set the STOP bit to generate the Stop condition. The interface automatically returns 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.
- **AF:** Detection of a non-acknowledge bit. In this case, the EVF and AF bits are set by hardware with an interrupt if the ITE bit is set. To resume, set the START or STOP bit.
- **ARLO:** Detection of an arbitration lost condition. In this case the ARLO bit is set by hardware (with an interrupt if the ITE bit is set and the interface automatically goes back to slave mode (the M/SL bit is cleared).

**Note:** In all these cases, the SCL line is not held low; however, the SDA line can remain low due to possible «0» bits transmitted last. Both lines must be released via software.

Figure 14.3 Transfer Sequencing

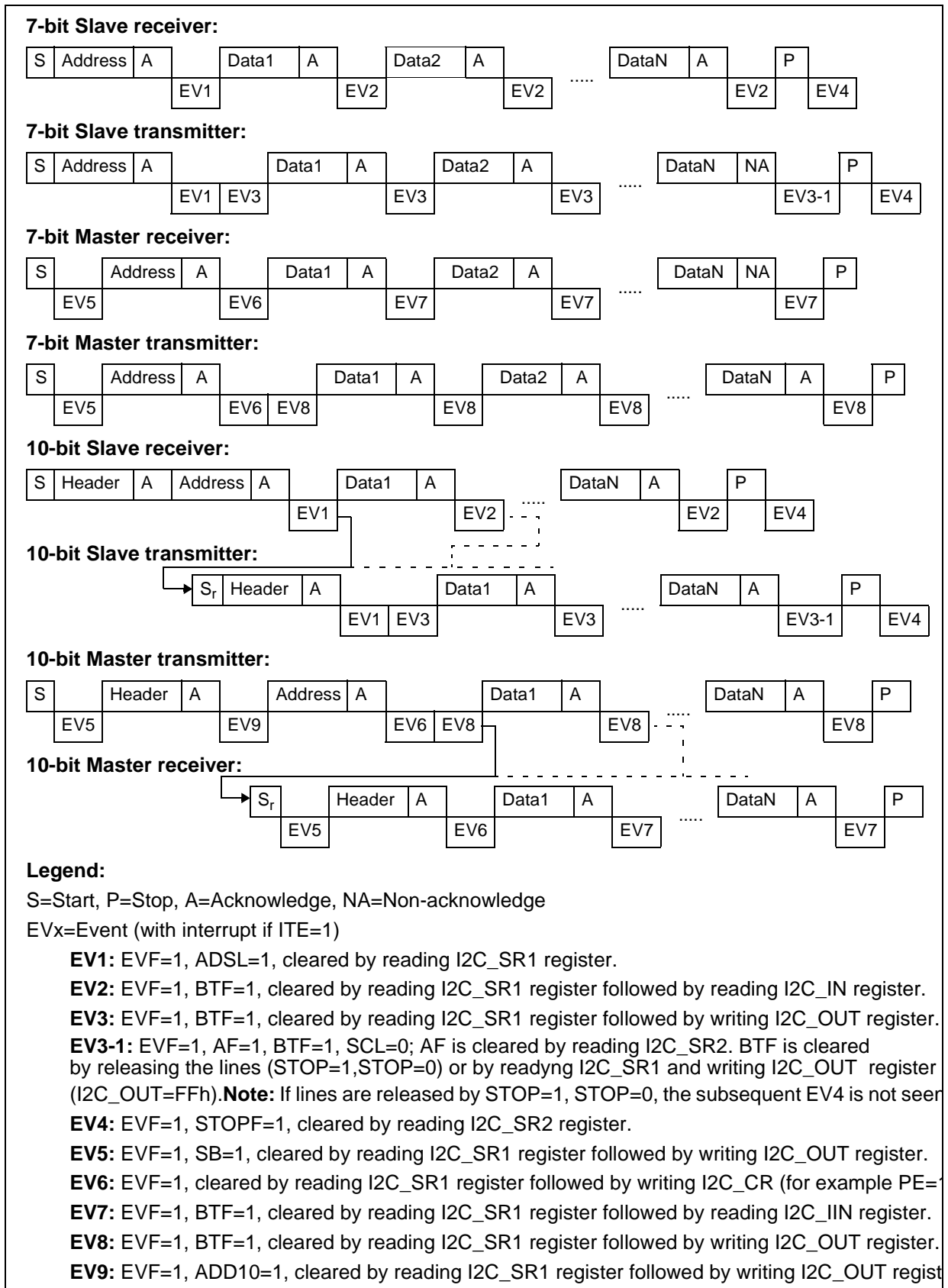
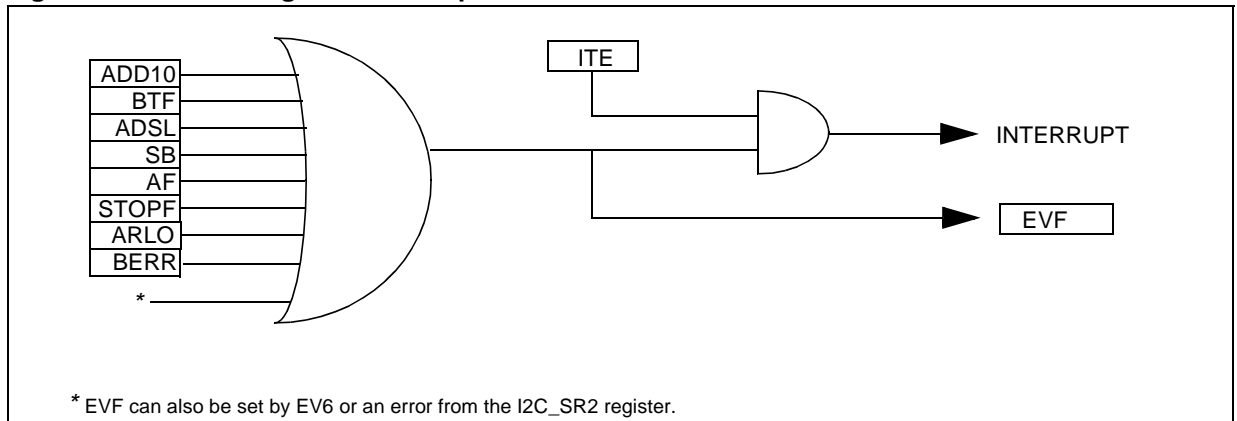


Figure 14.4 Event Flags and Interrupt Generation



Interrupt Event	Event Flag	Enable Control Bit	Exit from Wait	Exit from Halt
10-bit Address Sent Event (Master Mode)	ADD10	ITE	Yes	No
End of Byte Transfer Event	BTF		Yes	No
Address Matched Event (Slave Mode)	ADSEL		Yes	No
Start Bit Generation Event (Master Mode)	SB		Yes	No
Acknowledge Failure Event	AF		Yes	No
Stop Detection Event (Slave Mode)	STOPF		Yes	No
Arbitration Lost Event (Multimaster configuration)	ARLO		Yes	No
Bus Error Event	BERR		Yes	No

**Note:** The I<sup>2</sup>C interrupt events are connected to the same interrupt vector. They generate an interrupt if the corresponding Enable Control Bit (ITE) is set and the Interrupt Mask bit (MSKI2C) in the INT\_MASK Configuration Register is unmasked (set to 1, see Interrupts Chapter).

14.5 Register Description

In the following sections describe the registers used by the I<sup>2</sup>C Interface are described.

14.5.1 I<sup>2</sup>C Interface Configuration Registers.

I<sup>2</sup>C Control Register (I2C\_CR)

Configuration Register 16 (010h) Read/Write

Reset Value: 0000 0000 (00h)

7							0
-	-	PE	ENGC	START	ACK	STOP	ITE

Bit 7-6: Not Used. They must be held to 0.

Bit 5: **PE** Peripheral Enable.

This bit is set and cleared by software

0: peripheral disabled

1: peripheral enabled

**Notes:**

- When PE=0, all the bits of the I2C\_CR register and the SR register except the Stop bit are reset. All outputs are released while PE=0
- When PE=1, the corresponding I/O pins are selected by hardware as alternate functions.
- To enable the I<sup>2</sup>C interface, write the I2C\_CR register **TWICE** with PE=1 as the first write only activates the interface (only PE is set).

Bit 4: **ENGC** Enable General Call

This bit is set and cleared by software. It is also cleared by hardware when the interface is disabled (PE=0).

0: General Call disabled

1: General Call enabled

**Note:** The 00h General Call address is acknowledged (01h ignored).

Bit 3: **START** Generation of a Start Condition

This bit is set and cleared by software. It is also cleared by hardware when the interface is disabled (PE=0) or when the Start condition is sent (with interrupt generation if ITE=1).

- In Master Mode
  - 0: No Start generation
  - 1: Repeated Start generation

- In Slave Mode
  - 0: No Start generation
  - 1: Start generation when the bus is free

Bit 2: **ACK** Acknowledge enable

This bit is set and cleared by software. It is also cleared by hardware when the interface is disabled (PE=0).

0: No acknowledge returned

1: Acknowledge returned after an address byte or a data byte is received

Bit 1: **STOP** Reset signal mode

This bit is set and cleared by software. It is also cleared by hardware in master mode. Note: This bit is not cleared when the interface is disabled (PE=0).

- In Master Mode
  - 0: No Stop generation
  - 1: Stop generation after the current byte transfer or after the current Start condition is sent. The STOP bit is cleared by hardware when the Stop condition is sent.
- In Slave Mode
  - 0: No Start generation
  - 1: Release the SCL and SDA lines after the current byte transfer (BTF=1). In this mode the STOP bit has to be cleared by software.

Bit 0: **ITE** Interrupt Enable

0: Interrupt disabled

1: Interrupt enabled

I<sup>2</sup>C Clock Control Register (I2C\_CCR)

Configuration Register 17 (011h) Read/Write

Reset Value: 0000 0000 (00h)

7							0
FM/SM	CC6	CC5	CC4	CC3	CC2	CC1	CC0

Bit 7: **FM/SM** Fast/Standard I<sup>2</sup>C Mode.

This bit is set and cleared by software. It is not cleared when the interface is disabled (PE=0).

- 1: Standard I<sup>2</sup>C Mode (recommended up to 100 kHz)
- 0: Fast I<sup>2</sup>C Mode (recommended up to 400 kHz)

Bit 6-0: **CC6-CC0** 7-bit clock divider

These bits select the speed of the bus ( $F_{SCL}$ ) depending on the I<sup>2</sup>C mode. They are not cleared when the interface is disabled (PE=0). The speed can be computed as follows:

- Standard mode (FM/SM=1):  $F_{SCL} \leq 100\text{kHz}$   
 $F_{SCL} = f_{CPU}/(3x[CC6..CC0]+11)$
- Fast mode (FM/SM=0):  $F_{SCL} > 100\text{kHz}$   
 $F_{SCL} = f_{CPU}/(2x[CC6..CC0]+9)$

**Warning:** For safety reason, CC6-CC0 bits must be configured with a value  $\geq 3$  for the Standard mode and  $\geq 2$  for the Fast mode.

**I<sup>2</sup>C Own Address Register 1 (I2C\_OAR1)**

Configuration Register 18 (012h) Read/Write  
 Reset Value: 0000 0000 (00h)

7									0
ADD7	ADD6	ADD5	ADD4	ADD3	ADD2	ADD1	ADD0		

**7-bit Addressing Mode**

bit 7-1: **ADD7-ADD1** Interface address.

These bits define the I<sup>2</sup>C bus address of the interface. They are not cleared when the interface is disabled (PE=0).

Bit 0: **ADD0** Address direction bit.

This bit is “don’t care”, the interface acknowledges either 0 or 1. It is not cleared when the interface is disabled (PE=0).

**Note:** Address 01h is always ignored.

**10-bit Addressing Mode**

bit 7-0: **ADD7-ADD0** Interface address.

These are the least significant bits of the I<sup>2</sup>C bus address of the interface. They are not cleared when the interface is disabled (PE=0).

**I<sup>2</sup>C Own Address Register 2 (I2C\_OAR2)**

Configuration Register 19 (013h) Read/Write  
 Reset Value: 0000 0000 (00h)

7						2			0
-	-	-	-	-	-	ADD9	ADD8	-	-

Bit 7-3: Not Used

bit 7-1: **ADD8-ADD8** Interface address.

These are the most significant bits of the I<sup>2</sup>C bus address of the interface (10-bit mode only). They are not cleared when the interface is disabled (PE=0).

Bit 0: Reserved

**14.5.2 I<sup>2</sup>C Interface Input Registers.**

**I<sup>2</sup>C Data Input Register (I2C\_IN)**

Input Register 6 (06h) Read only  
 Reset Value: 0000 0000 (00h)

7									0
I2CDI7	I2CDI6	I2CDI5	I2CDI4	I2CDI3	I2CDI2	I2CDI1	I2CDI0		

bit 7-0: **I2CDI7-I2CDI0** Received data.

These bits contain the byte to be received from the bus in Receiver mode: the first data byte is received automatically in the I2C\_IN register using the least significant bit of the address.

Then, the next data bytes are received one by one after reading the I2C\_IN register.

**I<sup>2</sup>C Status Register 1 (I2C\_SR1)**

Input Register 7 (07h) Read only  
 Reset Value: 0000 0000 (00h)

7									0
EVF	ADD10	TRA	BUSY	BTF	ADSL	M/SL	SB		

**Bit 7: EVF Event Flag**

This bit is set by hardware as soon as an event occurs. It is cleared by software reading I2C\_SR2 register in case of error event or as described in Figure 14.3. It is also cleared by hardware when the interface is disabled (PE=0).

0: No event

1: One of the following events has occurred:

- BTF=1 (Byte received or transmitted)
- ADSL=1 (Address matched in Slave mode while ACK=1)
- SB=1 (Start condition generated in Master mode)
- AF=1 (No acknowledge received after byte transmission)
- STOPF=1 (Stop condition detected in Slave mode)
- ARLO=1 (Arbitration lost in Master mode)
- BERR=1 (Bus error, misplaced Start or Stop condition detected)
- Address byte successfully transmitted in Master mode.

**Bit 6: ADD10 10 bit addressing in Master Mode**

This bit is set by hardware when the master has sent the first byte in 10-bit address mode. It is cleared by software reading I2C\_SR2 register followed by a write in the I2C\_OUT register of the second address byte. It is also cleared by hardware when the peripheral is disabled (PE=0).

0: No ADD10 event occurred

1: The Master has sent the first address byte

**Bit 5: TRA Transmitter/Receiver**

When BTF is set, TRA=1 if a data byte has been transmitted. It is cleared automatically when BTF is cleared. It is also cleared by hardware after detection of Stop condition (STOPF=1), loss of bus arbitration (ARLO=1) or when the interface is disabled (PE=0).

0: Data byte received (if BTF=1)

1: Data byte transmitted

**Bit 4: BUSY Bus busy**

This bit is set by hardware on detection of a Start condition and cleared by hardware on detection of a Stop condition. It indicates a communication in progress on the bus. This information is still updated when the interface is disabled (PE=0).

0: No communication on the bus

1: Communication ongoing on the bus

**Bit 3: BTF Byte transfer finished**

This bit is set by hardware as soon as a byte is correctly received or transmitted with interrupt generation if ITE=1. It is cleared by software reading I2C\_SR1 register followed by a read of I2C\_IN or write of I2C\_OUT registers. It is also cleared by hardware when the interface is disabled (PE=0).

- Following a byte transmission, this bit is set after reception of the acknowledge clock pulse. In case an address byte is sent, this bit is set only after the EV6 event (see Figure 14.3). BTF is cleared by reading I2C\_SR1 register followed by writing the next byte in I2C\_OUT register.
- Following a byte reception, this bit is set after transmission of the acknowledge clock pulse if ACK=1. BTF is cleared by reading I2C\_SR1 register followed by reading the byte from I2C\_IN 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 slave address received 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 I2C\_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 I2C\_SR1 register followed by writing the address byte in I2C\_OUT register. It is also cleared by hardware when the interface is disabled (PE=0).

0: No Start condition

1: Start condition generated

### I<sup>2</sup>C Status Register 2 (I2C\_SR2)

Input Register 8 (08h) Read only

Reset Value: 0000 0000 (00h)

7							0
-	-	-	AF	STOPF	ARLO	BERR	GCAL

Bit 7-5: Reserved.

Bit 4: **AF** *Acknowledge failure.*

This bit is set by hardware when an acknowledge is returned. An interrupt is generated if ITE=1. It is cleared by software reading the I2C\_SR2 register or by hardware when the interface is disabled (PE=0).

The SCL line is not held low while AF=1.

0: No acknowledge failure

1: Acknowledge failure

Bit 3: **STOPF** *Stop detection (Slave mode).*

This bit is set by hardware when a Stop condition is detected on the bus after an acknowledge (if ACK=1). An interrupt is generated if ITE=1. It is cleared by software reading I2C\_SR2 register or by hardware when the interface is disabled (PE=0).

The SCL line is not held low while STOPF=1.

0: No Stop condition detected

1: Stop condition detected

Bit 2: **ARLO** *Arbitration lost.*

This bit is set by hardware when the interface loses the arbitration of the bus to another master. An interrupt is generated if ITE=1. It is cleared by software reading I2C\_SR2 register or by hardware when the interface is disabled (PE=0).

After an ARLO event the interface switches back automatically to Slave mode (M/SL=0).

The SCL line is not held low while ARLO=1.

0: No arbitration lost detected

1: Arbitration lost detected

Bit 1: **BERR** *Bus error.*

This bit is set by hardware when the interface detects a misplaced Start or Stop condition. An interrupt is generated if ITE=1. It is cleared by software reading I2C\_SR2 register or by hardware when the interface is disabled (PE=0).

The SCL line is not held low while BERR=1.

0: No misplaced Start or Stop condition

1: Misplaced Start or Stop condition

Bit 0: **GCAL** *General Call (Slave mode).*

This bit is set by hardware when a general call address is detected on the bus while ENGCG=1. It is cleared by hardware detecting a Stop condition (STOPF=1) or when the interface is disabled (PE=0).

0: No general call address detected on bus

1: general call address detected on bus

### 14.5.3 I<sup>2</sup>C Interface Output Registers.

#### I<sup>2</sup>C Data Output Register (I2C\_OUT)

Output Register 6 (06h) Read only

Reset Value: 0000 0000 (00h)

7							0
I2CDO7	I2CDO6	I2CDO5	I2CDO4	I2CDO3	I2CDO2	I2CDO1	I2CDO0

bit 7-0: **I2CDO7-I2CDO0** Data to be transmitted.

These bits contain the byte to be transmitted in the bus in Transmitter mode: Byte transmission start automatically when the software writes in the I2C\_OUT register.

## 15 SERIAL PERIPHERAL INTERFACE (SPI)

### 15.1 Introduction

The Serial Peripheral Interface (SPI) allows full-duplex, synchronous, serial communication with external devices. An SPI system may consist of a master, one or more slaves, or a system, in which devices may be either masters or slaves.

SPI is normally used for communication between the ICU and external peripherals or another ICU.

Refer to the Pin Description section in this datasheet for the device-specific pin-out.

### 15.2 Main Features

- Full duplex, three-wire synchronous transfers
- Master or slave operation
- Four master mode frequencies
- Maximum slave mode frequency = CKM/4.
- Four programmable master bit rates
- Programmable clock polarity and phase
- End of transfer interrupt flag
- Write collision flag protection
- Master mode fault protection capability.

### 15.3 General description

SPI is connected to external devices through 4 alternate pins:

- MISO: Master In / Slave Out pin
- MOSI: Master Out / Slave In pin
- SCK: Serial Clock pin
- $\overline{SS}$ : Slave select pin (if not done through software)

A basic example of interconnections between a single master and a single slave is illustrated in Figure 15.1

The MOSI pins are connected together as the MISO pins. In this manner, data is transferred serially between master and slave (most significant bit first).

When the master device transmits data to a slave device via the MOSI pin, the slave device responds by sending data to the master device via the MISO pin. This implies full duplex transmission with both data out and data in synchronized with the same clock signal (which is provided by the master device via the SCK pin).

The transmitted byte is replaced by the byte received and eliminates the need for separate transmit-empty and receiver-full bits. A status flag is used to indicate that the I/O operation is complete.

Four possible data/clock timing relationships may be chosen (see Figure 15.4), but master and slave must be programmed with the same timing mode.

### 15.4 Functional Description

Figure 15.2 shows the serial peripheral interface (SPI) block diagram.

This interface contains 3 dedicated registers:

- A Control Register (SPI\_CR)
- A Status Register (SPI\_STATUS\_CR)
- A Data Register for transmission (SPI\_OUT)
- A Data Register for reception (SPI\_IN)

#### 15.4.1 Master Configuration.

In a master configuration, the serial clock is generated on the SCK pin.

Figure 15.1 SPI Master Slave

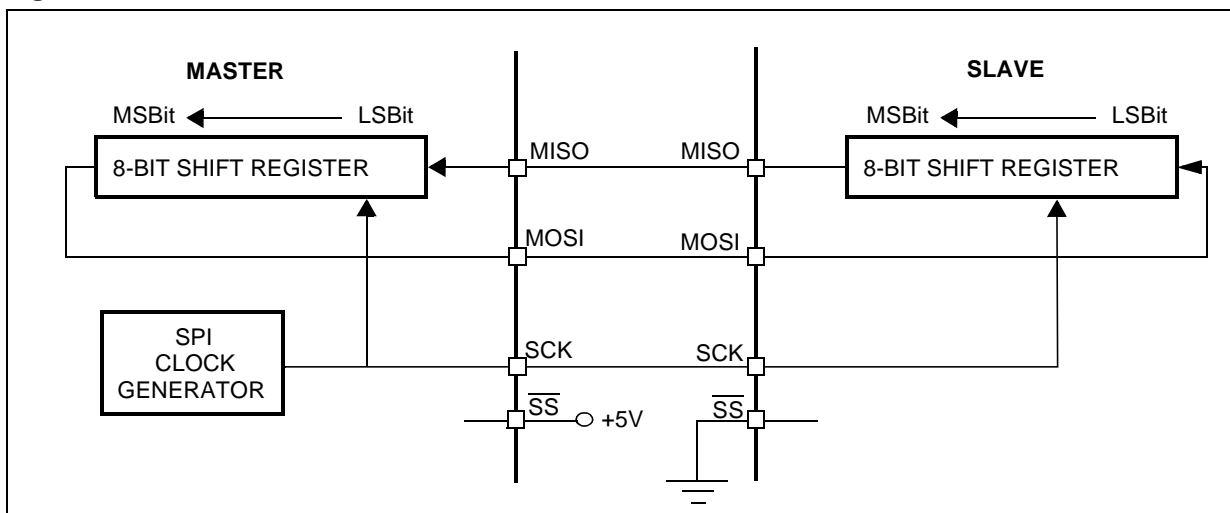
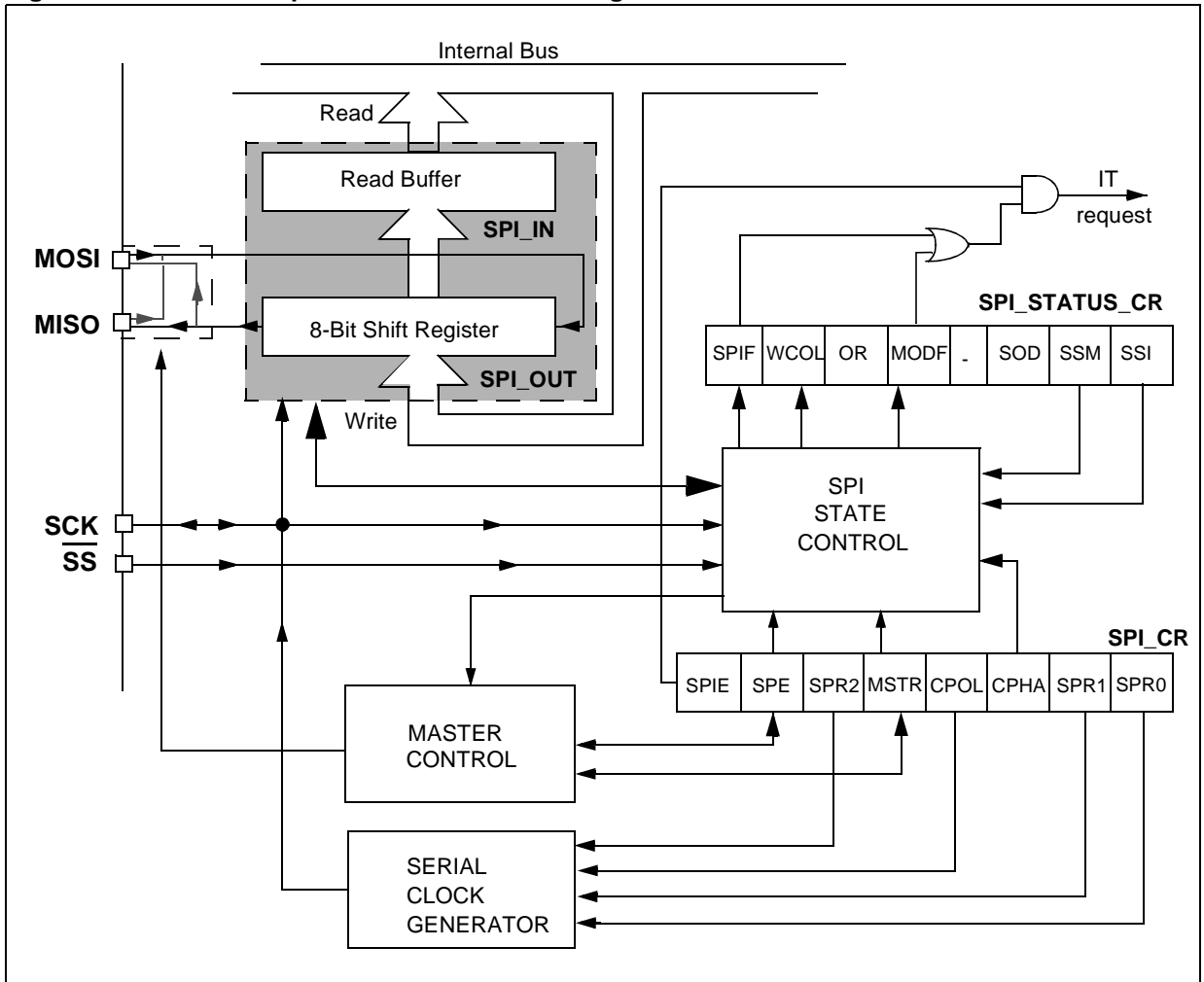




Figure 15.2 Serial Peripheral Interface Block Diagram



### Procedure

- Select the **SPR0**, **SPR1** and **SPR2** bits to define the serial clock baud rate (see **SPI\_CR** register).
- Select the **CPOL** and **CPHA** bits to define one of the four relationships between the data transfer and the serial clock (see Figure 15.4).
- The **SS** pin must be connected to a high level signal during the complete byte transmit sequence.
- The **MSTR** and **SPE** bits must be set (they remain set only if the **SS** pin is connected to a high level signal).

In this configuration the **MOSI** pin is a data output and to the **MISO** pin is a data input.

### Transmit sequence

Transmit sequence begins when a byte is written in the **SPI\_OUT** register.

The data byte is loaded in parallel into the 8-bit shift register (from the internal bus) during a write cycle 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 is generated if the **SPIE** bit is set.

During the last clock cycle the **SPIF** bit is set, a copy of the data byte received in the shift register is moved to a buffer. When the **SPI\_IN** register is read, the SPI peripheral returns this buffered value. Clearing the **SPIF** bit is performed by the following software sequence:

1. An access to the **SPI\_STATUS\_CR** register while the **SPIF** bit is set
2. A read to the **SPI\_IN** register.

**Note:** While the **SPIF** bit is set, all writes to the **SPI\_OUT** register are inhibited until the **SPI\_STATUS\_CR** register is read.

### 15.4.2 Slave Configuration.

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

The value of the SPR0, SPR1 and SPR2 bits is not used for data transfer.

#### Procedure

- For correct data transfer, the slave device must be in the same timing mode as the master device (CPOL and CPHA bits). See Figure 15.4.
- The  $\overline{SS}$  pin must be connected to a low level signal during the complete byte transmit sequence.
- Clear the MSTR bit and set the SPE bit to assign the pins to alternate function.

In this configuration the MOSI pin is a data input and the MISO pin is a data output.

#### Transmit Sequence

The data byte is loaded into the 8-bit shift register (from the internal bus) during a write cycle 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 is generated if SPIE bit is set.

During the last clock cycle the SPIF bit is set, a copy of the data byte received in the shift register is moved to a buffer. When the SPI\_IN register is read, the SPI peripheral returns the buffer value.

The SPIF bit is cleared by the following software sequence:

1. An access to the SPI\_STATUS\_CR register while the SPIF bit is set.
2. A read to the SPI\_IN register.

**Note:** While the SPIF bit is set, all writes to the SPI\_OUT register are inhibited until the SPI\_STATUS\_CR register is read.

The SPIF bit can be cleared during a second transmission; however, it must be cleared before the second SPIF bit in order to prevent an overrun condition (see Section 15.4.6).

Depending on the CPHA bit, the  $\overline{SS}$  pin has to be set to write to the SPI\_OUT register between each data byte transfer to avoid a write collision (see Section 15.4.4).

### 15.4.3 Data Transfer Format.

During an SPI transfer, data is simultaneously transmitted (shifted out serially) and received

(shifted in serially). The serial clock is used to synchronize data transfer during a sequence of eight clock pulses.

The  $\overline{SS}$  pin allows individual selection of a slave device; the other slave devices that are not selected do not interfere with SPI transfer.

#### Clock Phase and Clock Polarity

Four possible timing relationships may be chosen by software, using the CPOL and CPHA bits.

The CPOL (clock polarity) bit controls the steady state value of the clock when data isn't being transferred. This bit affects both master and slave modes.

The combination between the CPOL and CPHA (clock phase) bits select the data capture clock edge.

Figure 15.4, shows an SPI transfer with the four combinations of the CPHA and CPOL bits. The diagram may be interpreted as a master or slave timing diagram where the SCK pin, the MISO pin, the MOSI pin are directly connected between the master and the slave device.

The  $\overline{SS}$  pin is the slave device select input and can be driven by the master device.

The master device applies data to its MOSI pin-clock edge before the capture clock edge.

#### CPHA bit is set

The second edge on the SCK pin (falling edge if the CPOL bit is reset, rising edge if the CPOL bit is set) is the MSBit capture strobe. Data is latched on the occurrence of the second clock transition.

A write collision should not occur even if the  $\overline{SS}$  pin stays low during a transfer of several bytes (see Figure 15.3).

#### CPHA bit is reset

The first edge on the SCK pin (falling edge if CPOL bit is set, rising edge if CPOL bit is reset) is the MSBit capture strobe. Data is latched on the occurrence of the first clock transition.

The  $\overline{SS}$  pin must be toggled high and low between each byte transmitted (see Figure 15.3).

In order to protect the transmission from a write collision a low value on the  $\overline{SS}$  pin of a slave device freezes the data in its SPI\_OUT register and does not allow it to be altered. Therefore, the  $\overline{SS}$  pin must be high to write a new data byte in the SPI\_OUT without producing a write collision.

### 15.4.4 Write Collision Error.

A write collision occurs when the software tries to write to the SPI\_OUT register while a data transfer

is taking place with an external device. When this occurs, the transfer continues uninterrupted; and the software writing will be unsuccessful.

Write collisions can occur both in master and slave mode.

**Note:** a “read collision” will never occur since the data byte received is placed in a buffer, in which access is always synchronous with the ICU operation.

### In Slave mode

When the CPHA bit is set:

The slave device will receive a clock (SCK) edge prior to the latch of the first data transfer. This first clock edge will freeze the data in the slave device SPI\_OUT register and output the MSB bit on to the external MISO pin of the slave device.

The  $\overline{SS}$  pin low state enables the slave device, but the output of the MSB bit onto the MISO pin does not take place until the first data transfer clock edge occurs.

When the CPHA bit is reset:

Data is latched on the occurrence of the first clock transition. The slave device doesn't have a way of knowing when that transition will occur; therefore, the slave device collision occurs when software attempts to write the SPI\_OUT register after its  $\overline{SS}$  pin has been pulled low.

For this reason, the  $\overline{SS}$  pin must be high, between each data byte transfer, in order to allow the CPU to write in the SPI\_OUT register without generating a write collision.

### In Master mode

Collision in the master device is defined as a write of the SPI\_OUT register, while the internal serial clock (SCK) is in the process of transfer.

The  $\overline{SS}$  pin signal must always be high on the master device.

### WCOL bit

The WCOL bit in the SPI\_STATUS\_CR 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).

The WCOL bit is cleared by a software sequence (see Section 15.5).

### 15.4.5 Master Mode Fault.

Master mode fault occurs when the master device has its  $\overline{SS}$  pin pulled low, then the MODF bit is set.

Master mode fault affects the SPI peripheral in the following ways:

- The MODF bit is set and an SPI interrupt 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, forcing the device into slave mode.

Clearing the MODF bit is done through a software sequence:

1. A read or write access to the SPI\_STATUS\_CR register while the MODF bit is set.
2. A write to the SPI\_CR register.

**Note:** To avoid any multiple slave conflicts in the case of a system comprising several MCUs, the  $\overline{SS}$  pin must be pulled high during the clearing sequence of the MODF bit. The SPE and MSTR bits may be restored to their original state during or after this clearing sequence.

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't be set, but in a multi master configuration the device can be in slave mode with this MODF bit set.

The MODF bit indicates that there might have been a multi-master conflict for system control and allows a proper exit from system operation to a reset or default system state using an interrupt routine.

**Figure 15.3** CHPA/ $\overline{SS}$  Timing Diagram

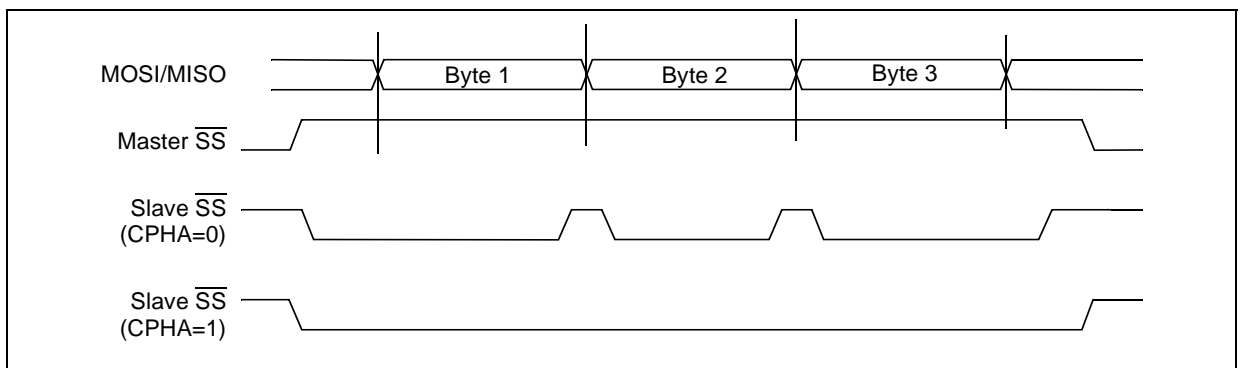


Figure 15.4 Data Clock Timing Diagram

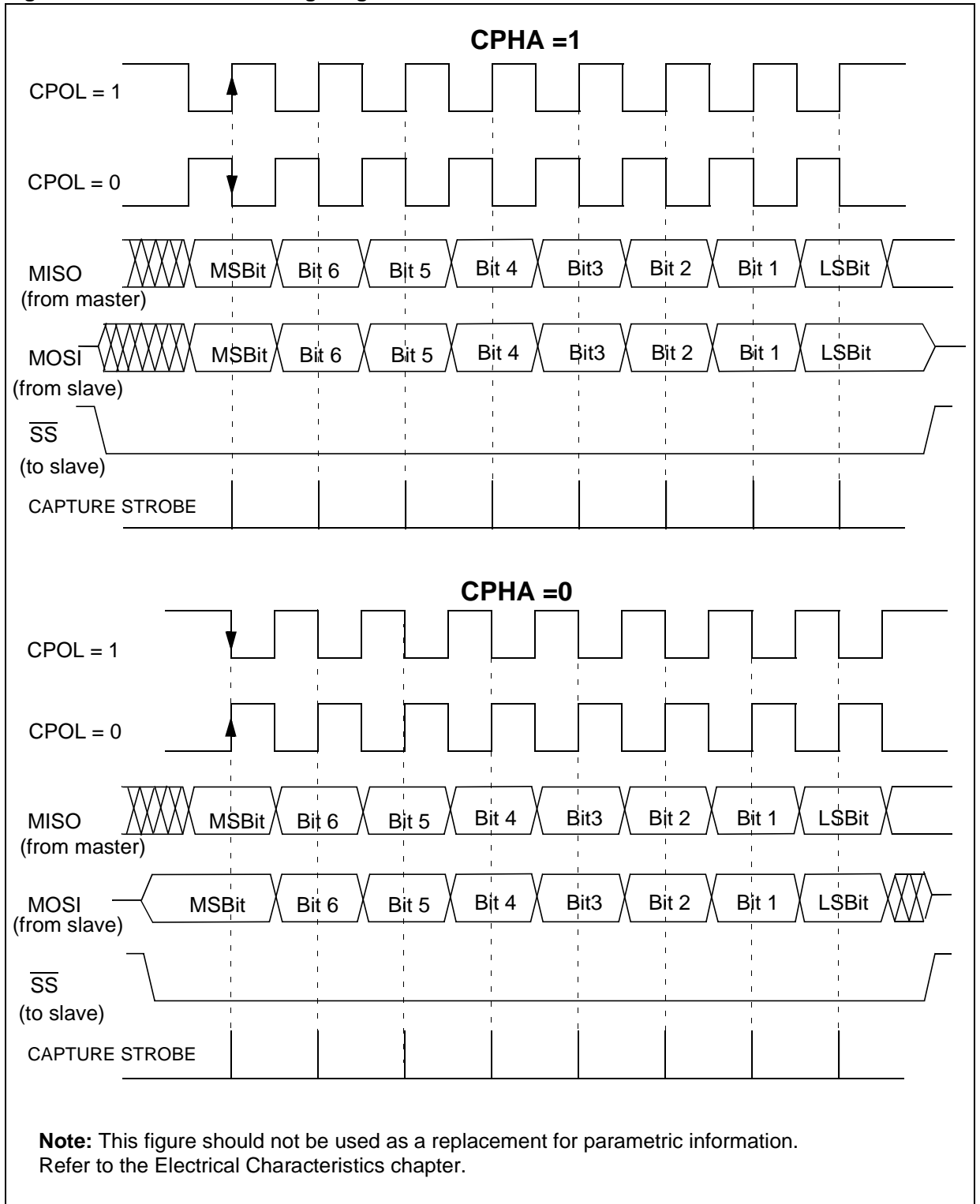
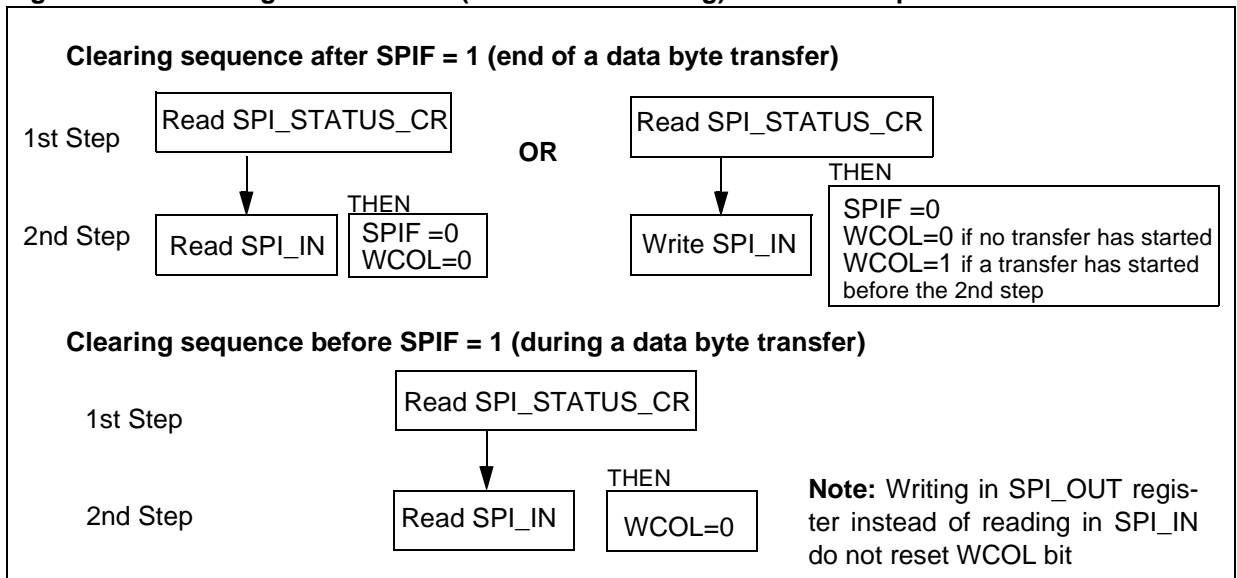


Figure 15.5 Clearing the WCOL bit (Write Collision Flag) Software Sequence



#### 15.4.6 Overrun Condition.

An overrun condition occurs when the master device has sent several data bytes and the slave device hasn't cleared the SPIF bit issued from the previous data byte transmitted.

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

This condition is not detected by the SPI peripheral.

#### 15.4.7 Single Master and Multimaster Configurations.

There are two types of SPI systems:

- Single Master System
- Multimaster System

##### Single Master System

A typical single master system may be configured, using an ICU as the master and four ICUs as slaves (see Figure 15.6).

The master device selects the individual slave devices by using four pins of a parallel port to control the four  $\overline{SS}$  pins of the slave devices.

The  $\overline{SS}$  pins are pulled high during reset since the master device ports will be forced to be inputs at that time, thus disabling the slave devices.

**Note:** In order to prevent a bus conflict on the MISO line the master allows only one active slave device during a transmission.

For more security, the slave device may respond to the master with the data byte received. Then the master will receive the previous byte back from the slave device if all MISO and MOSI pins are connected and the slave has not written its SPI\_OUT register.

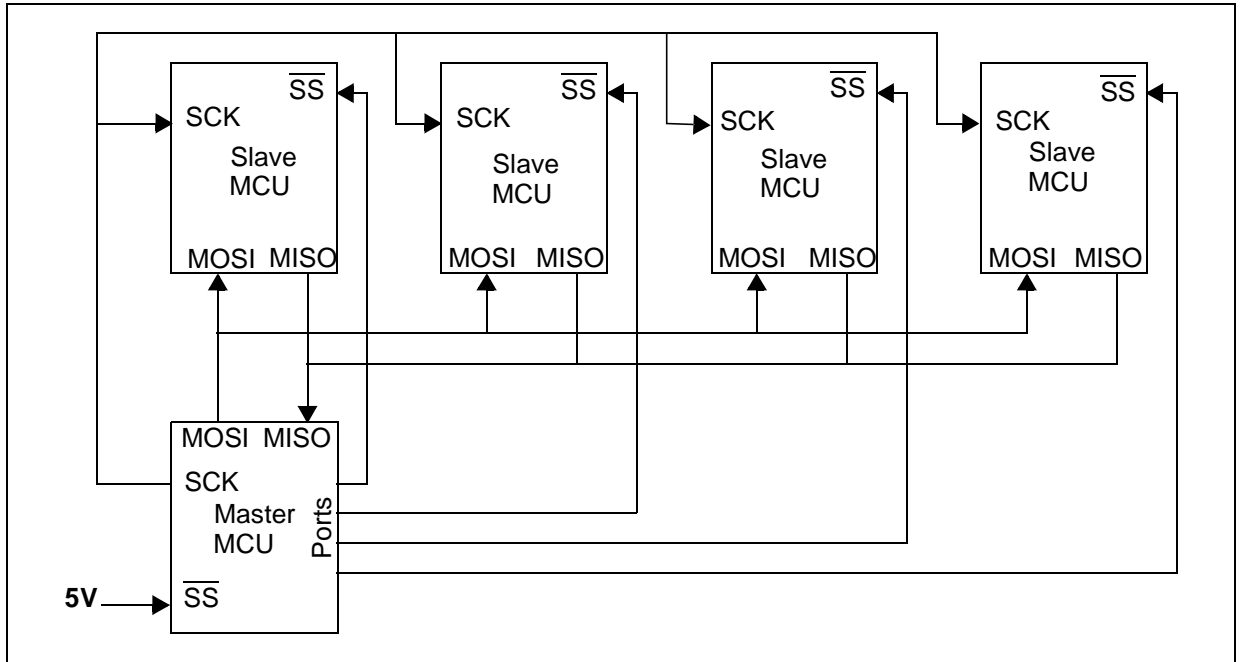
Other transmission security methods can use ports for handshake lines or data bytes with command fields.

##### Multi-master System

A multi-master system may also be configured by the user. Transfer of master control could be implemented using a handshake method through the I/O ports or by an exchange of code messages through the serial peripheral interface system.

The multi-master system is principally handled by the MSTR bit in the SPI\_CR register and the MODF bit in the SPI\_STATUS\_CR register.

Figure 15.6 Single Master Configuration



15.4.8 Interrupts

Interrupt Event	Event Flag	Enable Control Bit	Exit from Wait	Exit from Halt
SPI End of Transfer Event	SPIF	SPIE	Yes	No
Master Mode Fault Event	MODF		Yes	No

**Note:** The SPI interrupt events are connected to the same interrupt vector (see Interrupts chapter). They generate an interrupt if the corresponding Enable Control Bit (SPIE) and the interrupt mask bit (MSKSPI) in the INT\_MASK Configuration Register is set.

### 15.5 SPI Register Description

In the following sections describe the registers used by the SPI.

#### 15.5.1 SPI Configuration Registers.

##### SPI Control Register (SPI\_CR)

Configuration Register 20 (014h) Read/Write

Reset Value: 0000 0000 (00h)

7							0
SPIE	SPE	SPR2	MSTR	CPOL	CPHA	SPR1	SPR0

Bit 7: **SPIE** Serial peripheral interrupt enable.

This bit is set and cleared by software.

0: Interrupt is inhibited

1: An SPI interrupt is generated whenever SPIF=1 or MODF=1 in SPI\_STATUS\_CR

Bit 6: **SPE** Serial peripheral output enable.

This bit is set and cleared by software. It is also cleared by hardware when, in master mode, SS=0 (see Section 15.4.5 Master Mode Fault).

0: I/O port connected to pins

1: SPI alternate functions connected to pins

**Note:** The SPE bit is cleared by reset, so the SPI peripheral is not initially connected to the pins.

Bit 5: **SPR2** Divider Enable.

This bit is set and cleared by software and it is cleared by reset. It is used with the SPR[1:0] bits to set the baud rate. Refer to Table 15.1.

0: Divider by 2 enabled

1: Divider by 2 disabled

**Note:** This bit has no effect in slave mode.

Bit 4: **MSTR** Master/Slave mode select.

This bit is set and cleared by software. It is also cleared by hardware when, in master mode, SS=0 (see Section 15.4.5 Master Mode Fault).

0: Slave mode is selected

1: Master mode is selected, the function of the SCK pin changes from an input to an output and the functions of the MISO and MOSI pins are reversed.

Bit 3: **CPOL** Clock polarity.

This bit is set and cleared by software. This bit determines the steady state of the serial Clock. The CPOL bit affects both the master and slave modes.

0: The steady state is a low value at the SCK pin.

1: The steady state is a high value at the SCK pin.

**Note:** SPI must be disabled by resetting the SPE bit if CPOL is changed at the communication byte boundaries.

Bit 2: **CPHA** Clock phase.

This bit is set and cleared by software.

0: The first clock transition is the first data capture edge.

1: The second clock transition is the first capture edge.

Bit 1-0: **SPR1-SPR0** Serial peripheral rate.

These bits are set and cleared by software. Used with the SPR2 bit, they select one of six baud rates to be used as the serial clock when the device is a master (see Table 15.1).

These 2 bits have no effect in slave mode.

**Remark:** It is recommended to write the SPI\_CR register after the SPI\_STATUS\_CR register when working in master mode, vice versa when working in slave mode.

**Table 15.1 Serial Peripheral Baud Rate**

Serial Clock	SPR2	SPR1	SPR0
$f_{CKM}/2$	1	0	0
$f_{CKM}/4$	0	0	0
$f_{CKM}/8$	0	0	1
$f_{CKM}/16$	1	1	0
$f_{CKM}/32$	0	1	0
$f_{CKM}/64$	0	1	1

##### SPI Control-Status Register (SPI\_STATUS\_CR)

Configuration Register 21 (015h) Read/Write

Reset Value: 0000 0000 (00h)

7							0
SPIF	WCOL	OR	MODF	-	SOD	SSM	SSI

Bit 7: **SPIF** Serial Peripheral data transfer flag. (read only)

This bit is set by hardware when a transfer has been completed. An interrupt is generated if SPIE=1 in the SPI\_CR register. It is cleared by a software sequence (an access to the SPI\_STATUS\_CR register followed by a read or write to the SPI\_IN/SPI\_OUT registers).

- 0: Data transfer is in progress or has been approved by a clearing sequence.
- 1: Data transfer between the device and an external device has been completed.

**Note:** While the SPIF bit is set, all writes to the SPI\_OUT register are inhibited.

Bit 6: **WCOL** Write Collision status (read only).

This bit is set by hardware when a write to the SPI\_OUT register is done during a transmit sequence. It is cleared by a software sequence (see Figure 15.5).

- 0: No write collision occurred
- 1: A write collision has been detected

Bit 5: **OR** SPI overrun error (read only).

This bit is set by hardware when the byte currently being received in the shift register is ready to be transferred into the SPI\_IN register while SPIF = 1 (See Section 15.4.6 Overrun Condition). It is cleared by a software sequence (read of the SPI\_STATUS\_CR register followed by a read in SPI\_IN or write of the SPI\_OUT register).

- 0: No overrun error.
- 1: Overrun error detected.

Bit 4: **MODF** Mode Fault flag (read only).

This bit is set by hardware when the  $\overline{SS}$  pin is pulled low in master mode (see Section 15.4.5 Master Mode Fault). An SPI interrupt can be generated if SPIE=1 in the SPI\_CR register. This bit is cleared by a software sequence (An access to the SPI\_STATUS\_CR register while MODF=1 followed by a write to the SPI\_CR register).

- 0: No master mode fault detected
- 1: A fault in master mode has been detected

Bit 3: Not used.

Bit 2: **SOD** SPI output disable

This bit is set and cleared by software. When set, it disables the alternate function of the SPI output (MOSI in master mode / MISO in slave mode)

- 0: SPI output not disable
- 1: SPI output disable.

Bit 1: **SSM**  $\overline{SS}$  mode selection

This bit is set and cleared by software. When set, it disables the alternate function of the SPI Slave Select pin and use the SSI bit value instead of.

- 0:  $\overline{SS}$  pin used by the SPI.
- 1:  $\overline{SS}$  pin not used (I/O mode), SSI bit value is used.

Bit 0: **SSI**  $\overline{SS}$  internal mode

This bit is set and cleared by software. It replaces pin  $\overline{SS}$  of the SPI when bit SSM is set to 1. SSI bit is active low slave select signal when SSM is set to 1.

- 0 : Slave selected
- 1 : Slave not selected.

**Remark:** It is recommended to write the SPI\_CR register after the SPI\_STATUS\_CR register when working in master mode, vice versa when working in slave mode.

**15.5.2 SPI Input Register.**

**SPI Data Input Register (SPI\_IN)**

Input Register 5 (05h) Read only  
Reset Value: 0000 0000 (00h)

7							0
SPIDI7	SPIDI6	SPIDI5	SPIDI4	SPIDI3	SPIDI2	SPIDI1	SPIDI0

bit 7-0: **SPIDI7-SPIDI0** Received data.

The SPI\_IN register is used to receive data on the serial bus.

**Note:** During the last clock cycle the SPIF bit is set, a copy of the data byte received in the shift register is moved to a buffer. When the user reads the serial peripheral data I/O register, the buffer is actually being read.





**Warning:** A read to the *SPI\_IN* register returns the value located in the buffer and not the contents of the shift register (see Figure 15.2).

### 15.5.3 SPI Output Register.

#### SPI Data Output Register (SPI\_OUT)

Output Register 5 (05h) Write only

Reset Value: 0000 0000 (00h)

7							0
SPIDO7	SPIDO6	SPIDO5	SPIDO4	SPIDO3	SPIDO2	SPIDO1	SPIDO0

bit 7-0: **SPIDO7-SPIDO0** Data to be transmitted.

The SPI\_OUT register is used to transmit data on the serial bus. In the master device only a write to this register will initiate transmission/reception of another byte.

**Warning:** A write to the *SPI\_OUT* register places data directly into the shift register for transmission.

Full Product Information at <http://mcu.st.com>

Information furnished is believed to be accurate and reliable. However, STMicroelectronics assumes no responsibility for the consequences of use of such information nor for any infringement of patents or other rights of third parties which may result from its use. No license is granted by implication or otherwise under any patent or patent rights of STMicroelectronics. Specification mentioned in this publication are subject to change without notice. This publication supersedes and replaces all information previously supplied. STMicroelectronics products are not authorized for use as critical components in life support devices or systems without express written approval of STMicroelectronics.

The ST logo is a registered trademark of STMicroelectronics

© 2003 STMicroelectronics – Printed in Italy – All Rights Reserved

STMicroelectronics GROUP OF COMPANIES

Australia - Brazil - China - Canada - Finland - France - Germany - Hong Kong - India - Israel - Italy - Japan - Malaysia - Malta  
- Morocco - Singapore - Spain - Sweden - Switzerland - United Kingdom - U.S.A.

<http://www.st.com>