Introduzione ai uC **ATMEL©** *– Serie* **AVR** *e ai linguaggi di programmazione ad alto livello*

Giovanni De Luca

Laboratorio Progettazione Elettronica

www.delucagiovanni.com
deluca@lns.infn.it

Scopo del corso

- *Il corso propone lo studio del core AVR della ATMEL con particolare riferimento al dispositivo ATMEGA128 e alle sue periferiche di I/O.*
- *L'ambiente di sviluppo utilizzato è il BASCOM-AVR della MCS-Electronics e si avvale dell'ausilio di AVRStudio4 di ATMEL per il debug e di PROTEUS-ISIS per l'emulazione software.*
- *Il corso è rivolto a Tecnici elettronici, progettisti dell'area elettronica digitale e a studenti di scuole ad indirizzo elettronico*
- *Requisiti : Buona conoscenza del sistema operativo Windows, dell'elettronica digitale e della strumentazione elettronica di base.*

Concetti essenziali e pratici

- *Il corso si baserà su concetti essenziali e pratici per portare l'utilizzatore neofita a realizzare nel più breve tempo possibile hardware embedded basato su microcontrollori RISC della famiglia AVR.*

- *Anche chi non ha mai visto un uC alla fine del corso dovrà aver imparato a giocare con questi oggettini neri.*

Alcuni Argomenti trattati nel corso

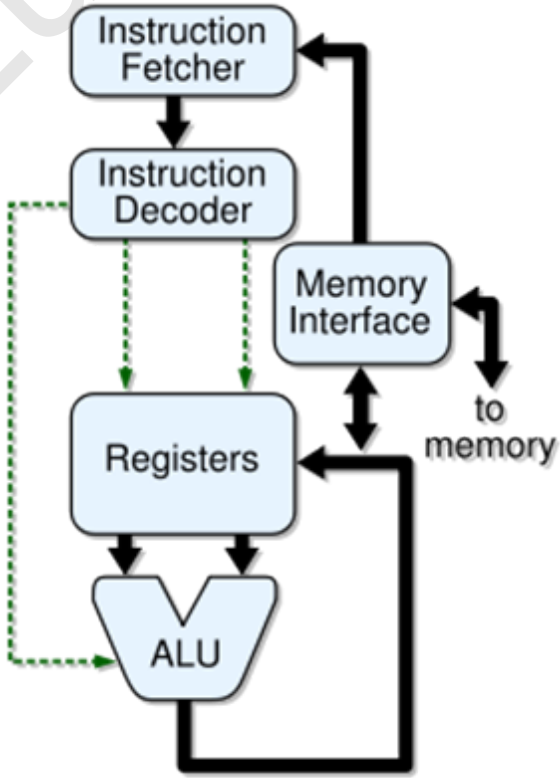
- *Concetti base sui microprocessori e microcontrollori*
- *Architettura di un microcontrollore RISC Atmel Atmega128 Harvard*
- *Descrizione dell'architettura e dei registri di un microcontrollore AVR tipico*
- *Descrizione di AVR CPU CORE*
- *Descrizione sommaria del set di istruzioni assembly*
- *Descrizione e uso dei principali comandi del linguaggio Bascom-AVR*
- *Gestione della memoria, delle variabili e delle costanti in un programma FW*
- *Utilizzo della memoria EEPROM interna*
- *Utilizzo delle librerie e uso del compilatore BAS-AVR*
- *Gestione e configurazione del Clock di sistema*

..... continua

- *Gestione e configurazione degli interrupts tramite i comandi BASCOM*
- *Gestione e configurazione delle porte di IO digitali*
- *Gestione e impostazione dei Timer e del PWM*
- *Utilizzo delle porte di comunicazione seriale UART*
- *Utilizzo della USART per comunicazioni I2C & TWI Two Wire e 1Wire Interface*
- *Gestione della porta SPI Serial Peripheral Interface*
- *Gestione e configurazione del convertitore interno ADC*
- *Metodi di programmazione e struttura di un firmware per un controllo automatico*
- *Esempi pratici per gestire LCD e periferiche di I/O (keyboard e IO)*
- *Metodi di accesso a memorie e periferiche esterne*
- *Esercitazioni pratiche con l'ausilio di un simulatore/emulatore hardware*

Cenni sul uProcessore

- *Un microprocessore è un singolo circuito integrato in grado di effettuare operazioni di calcolo o di elaborazione dell'informazione.*
- *Il microprocessore principale di un computer viene chiamato comunemente processore o CPU.*
- *La **CPU**, acronimo dell'inglese "Central Processing Unit", traduzione letterale **unità centrale di elaborazione**, anche chiamata nella sua implementazione fisica **processore**, è uno dei due componenti della macchina di **Turing** (l'altro è la memoria).*
- *Compito della CPU è quello di leggere le istruzioni e i dati dalla memoria ed eseguire le istruzioni; il risultato della esecuzione di una istruzione dipende dal dato su cui opera e dallo stato interno della CPU stessa, che tiene traccia delle passate operazioni.*

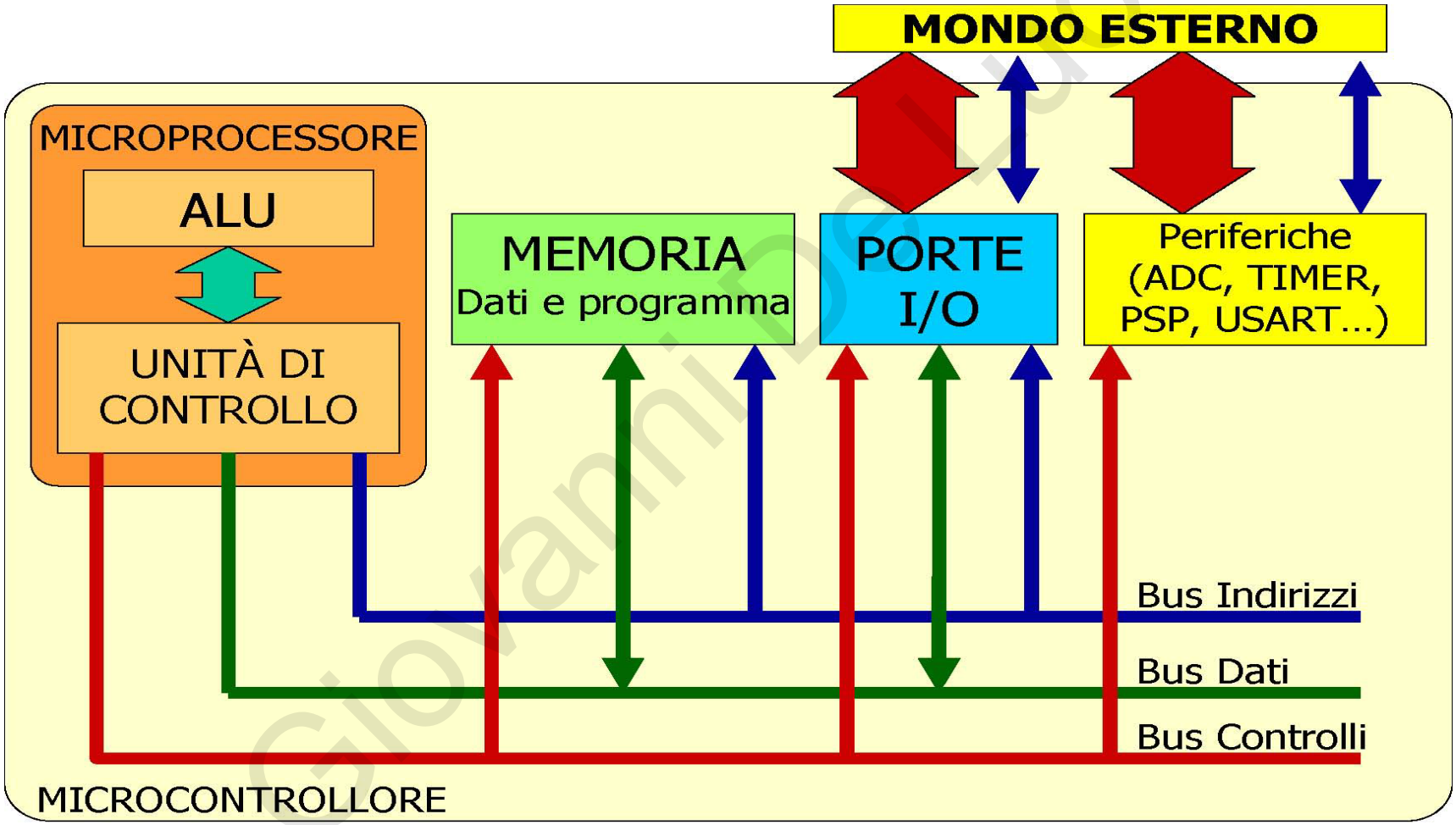


Cosa è un Microcontrollore

- *Un microcontrollore o microcontroller, detto anche computer single chip è un sistema a microprocessore completo, integrato in un solo chip, progettato per ottenere la massima autosufficienza funzionale ed ottimizzare il rapporto prezzo-prestazioni per una specifica applicazione, a differenza, ad esempio, dei microprocessori impiegati nei personal computer, adatti per un uso più generale.*

- *I microcontroller sono la forma più diffusa e più invisibile di computer. Comprendono la CPU, un certo quantitativo di memoria RAM e memoria ROM (può essere PROM, EPROM, EEPROM o FlashROM) e una serie di interfacce di I/O (input/output) standard, fra cui bus (I^2C, SPI, CAN). Le periferiche integrate sono la vera forza di questi dispositivi: si possono avere convertitori ADC e convertitori DAC multicanale, timer/counter, USART, numerose porte esterne bidirezionali bufferizzate, comparatori, PWM.*

Un microcontrollore tipico



Le periferiche principali di un microcontrollore

Contiene programmi e dati residenti	**ROM**	**RAM**	Contiene dati temporanei
Esegue le istruzioni e gestisce eventi	**CPU**	**EEPROM**	Contiene dati permanenti
Generano eventi, contatori e operano come base di tempi	**TIMER**	**ADC**	Per l'acquisizione di segnali analogici
Per l'interfacciamento con le periferiche	**I/O PORTS**	**USART**	Per la ricezione e la trasmissione di dati

Control Bus e Data Bus

- ***BUS.*** *Si indica con il termine **bus** l'insieme dei collegamenti fisici tra CPU e ogni altro blocco funzionale presente nel calcolatore (memoria, periferiche, dispositivi di I/O)*
- *Su un bus possono circolare: dati, indirizzi di locazioni di memoria o segnali di controllo.*

- ***CONTROL BUS****: è l'insieme di tutti i segnali di controllo scambiati dalla CPU con i dispositivi di memoria, di elaborazione dati, di input/output che coordinano e sincronizzano le attività dell'intero sistema.*

- ***ADDRESS BUS****: trasporta gli indirizzi (univoci) corrispondenti ad una locazione di memoria o a un dispositivo di input/output.*

- ***DATA BUS****: trasporta dati memorizzati in una locazione di memoria (indirizzata preventivamente grazie al bus indirizzi) alla CPU o viceversa.*

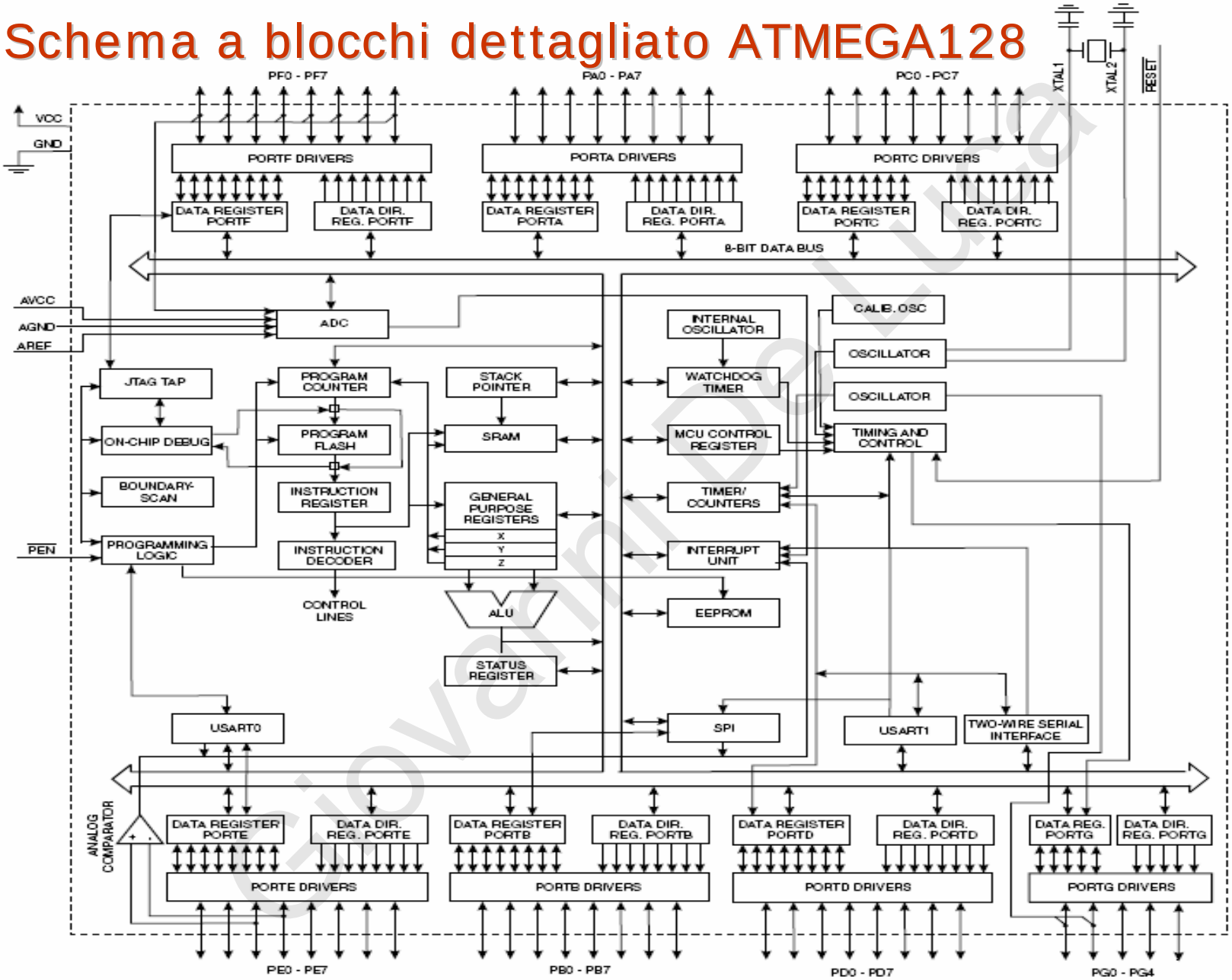
Le Memorie

- *La **ROM FLASH**(Read Only Memory) è solitamente utilizzata per la memorizzazione delle istruzioni del programma (può essere permanente o non permanente).*
- *Nella **RAM** (Random Access Memory) vengono custoditi i dati prodotti e impiegati durante l'esecuzione del programma.*
- *La **EEPROM** (Electrical Erasable Programmable Read Only Memory) serve per memorizzare i dati che devono essere conservati anche al termine del programma.*
- *La **CPU** (Central Processing Unit) è l'unità che, comunicando con le varie periferiche interne attraverso i **BUS**, si fa carico di eseguire il programma ed elaborare i dati. Per esempio, l'esecuzione di un programma consiste nella lettura sequenziale da parte della CPU delle istruzioni memorizzate nella memoria di programma.*

Funzioni HW del micro AVR

- *Interrupts dedicati*
- *Watchdog programmabile*
- *Timer / Counters / PWM*
- *Serial communication USART,SPI,TWI,USB*
- *ADC e Analog Interfaces*
- *J-TAG Interface e Debug*

Schema a blocchi dettagliato ATMEGA128



Interrupts

- *Il flusso di un programma e sempre di tipo sequenziale e può essere alterato da particolari istruzioni che volutamente intendono deviare il flusso del programma stesso. Gli interrupt forniscono invece un meccanismo di "congelamento" del flusso del programma in corso, eseguono una subroutine e ripristinano il normale funzionamento del programma come se nulla fosse accaduto. Questa subroutine,denominate "interrupt handler" viene eseguita soltanto quando si verifica un determinato evento che attiva l'interrupt. L'evento può essere: il timer che va in overflow, la ricezione di un byte dalla seriale, la fine della trasmissione di un byte dalla seriale e uno o due eventi esterni. L'AVR può essere configurato per gestire un interrupt handler per ognuno di questi eventi.*
- *La capacità di interrompere la normale esecuzione di un programma quando un particolare evento si verifica, rende il programma stesso molto più efficiente nel gestire processi asincroni.*

ADC e WATCH DOG

- *L'**ADC** (convertitore Analogico/digitale) converte un segnale esterno analogico (prelevato tipicamente in tensione) in una sua rappresentazione digitale. I microcontrollori che implementano tali dispositivi sono particolarmente adatti per essere impiegati in applicazioni di controllo.*

- *Il **WATCHDOG TIMER,** se attivato, genera un reset ad un intervallo prestabilito in fase di programmazione. Questo riavvio ciclico è utile per far uscire il microcontrollore da eventuali situazioni di stallo.*

Timers e I/O ports

- *I **TIMER** consentono al sistema di misurare e sincronizzare sia eventi interni che esterni. Sono impiegati anche per la generazione di segnali esterni di controllo.*

- *Le **I/O PORTS** sono le porte di ingresso ed uscita solitamente utilizzate per acquisire dati o per pilotare componenti. Usualmente, una porta I/O è costituita da 8 pin (anche meno qualche volta), programmabili sia come ingressi che uscite.*

Seriali

- *Una delle caratteristiche più potenti dell'AVR e' la UART integrata, conosciuta anche come porta seriale. Questo vuol dire che si può facilmente leggere e scrivere dati sulla porta seriale.*

- *L'unica operazione da effettuare è quella di configurare la porta seriale con il modo operativo ed il baud rate. Una volta configurata la porta è sufficiente leggere o scrivere un registro per ricevere od inviare dei dati in linea. L' AVR dal suo canto ci avvertirà automaticamente quando avrà finito di trasmettere un byte oppure quando avrà terminato di ricevere un byte.*

- *Le **INTERFACCE SERIALI** sono utilizzate per scambiare dati con il mondo esterno. Sono frequenti periferiche di comunicazione sia sincrone (SPI, I2C) che asincrone (USART). Le prime sono utilizzate solitamente per la comunicazione con dispositivi esterni (memorie, sensori), le seconde per comunicare con PC o altri µC (comunicazione seriale RS232 o RS422/485).*

PWM

- *Il **PULSE WIDTH MODULATION SIGNAL GENERATOR** è un dispositivo utilizzato per la generazione di segnali modulati in ampiezza d'impulso. È anche impiegato nella conversione DA per la generazione di segnali analogici facendo transitare il segnale in un filtro passa basso. Se la costante di tempo del filtro è scelta opportunamente, è possibile generare un segnale proporzionale in tensione al duty-cycle*.

J-Tag

- ***JTAG**, acronimo di **J**oint **T**est **A**ction **G**roup, è un consorzio di 200 imprese produttrici di circuiti integrati e circuiti stampati allo scopo di definire un protocollo standard per il test funzionale di tali dispositivi, che tendono ad essere sempre più complessi e difficili da controllare, fino a rendere impraticabili i tradizionali metodi manuali o automatici (e ad aumentare in modo non competitivo il "**Time to market**").*

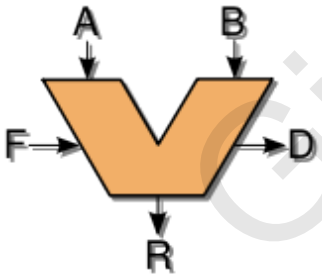
- *Consapevoli di queste esigenze le grandi industrie hanno scelto una strada che, sacrificando in piccola parte le risorse disponibili nel circuito integrato, permetta un collaudo razionale della scheda su cui viene installato. Oltre a questo vi è la possibilità di avere delle funzionalità aggiuntive (rivolgendo il controllo dai piedini verso l'interno del componente) quali la facile programmazione di memorie, microcontrollori (e altri dispositivi programmabili) con un'unica operazione, la possibilità di debug del firmware.*

ALU

- *L'Arithmetic Logic Unit (ALU) è un'unità dedita allo svolgimento di operazioni matematiche ed è formata da un insieme di porte logiche opportunamente collegate. Queste porte logiche nel loro complesso provvedono ad eseguire tutte le operazioni aritmetiche e logiche gestite dal microprocessore.*

- *Le prime ALU erano in grado di eseguire nativamente solo le operazioni più semplici (addizione, sottrazione e shifting di bit ecc.) e le operazioni logiche booleane (AND, OR, XOR e NOT).*

  Le operazioni più complesse come le operazioni di moltiplicazioni o divisione venivano emulate utilizzando ripetutamente somme o sottrazioni. Con l'evolvere dell'elettronica si è riuscito a integrare nelle ALU anche le operazioni di divisione e moltiplicazione.



Simbolo di una ALU.
A e **B** sono i segnali con gli addendi,
R sono i segnali con il risultato,
F sono i segnali di controllo dell'ALU e
D è il segnale con l'eventuale riporto/resto

Architettura RISC

- *RISC è l'acronimo di **R**educed **I**nstruction **S**et **C**omputer. Abbiamo circa sessanta o settanta istruzioni elementari (logiche, aritmetiche e istruzioni di trasferimento memoria-registro e registro-registro);*

  nella famiglia AVR arrivano a 130/140 : hanno tutte lo stesso formato e la stessa lunghezza, e molte vengono eseguite in un solo ciclo di clock. I processori RISC posseggono una unità di controllo semplice e a bassa latenza, riservando invece molto spazio per i registri interni:
- *una CPU RISC ha di solito da un minimo di una decina ad alcune migliaia di registri interni generici, organizzati in un file di registri. Il fatto di avere un formato unico di istruzione permette di strutturare l'unità di controllo come una pipeline, cioè una catena di montaggio a più stadi: questa innovazione ha il grosso vantaggio di ridurre il critical path interno alla CPU e consente ai RISC di raggiungere frequenze di clock più alte rispetto agli analoghi CISC.*

Architettura di un microcontrollore RISC Atmel Atmega128 Harvard

- *L' architettura **AVR ATMEL (Advanced RISC)** è del tipo "**RISC Harvard**" e dispone di un ricco ed efficiente set di istruzioni con 32 registri general purpose che sono connessi direttamente all' ALU in modo da renderne accessibili due contemporaneamente in un singolo ciclo di clock e poter raggiungere velocità di esecuzione fino a 10 volte superiori rispetto ai tradizionali microcontrollori CISC (Complex Instruction Set Computer) tipo 8051*
- *All'inizio i processori venivano progettati seguendo la classica architettura di **Von Neumann**. Secondo questa architettura la memoria del computer era vista come un nastro infinito e il processore era una testina che leggeva sequenzialmente i dati sul nastro, li elaborava e si spostava sul nastro di conseguenza.*

Architettura di un microcontrollore RISC Atmel Atmega128 Harvard

- *L'architettura di **Von Neumann** risultava inefficiente nella gestione di più flussi di dati essendo il flusso delle operazioni e dei dati mischiati.*

  *Per superarne i limiti venne sviluppata l'architettura **Harvard**. Questa architettura prevede che il flusso dati e il flusso delle istruzioni viaggino su due canali (BUS) separati all'interno del processore in modo da non disturbarsi a vicenda. Praticamente tutti i moderni processori sono basati su questa architettura dato che la separazione dei dati e delle istruzioni permette agli algoritmi che gestiscono le cache dei processori di funzionare al meglio.*
- *La famiglia di microcontrollori AVR della Atmel sono stati concepiti secondo questa architettura.*

Cosa è un ATMEGA

- *L' **ATMEGA128** è un microcontrollore CMOS a 8-bit basato sull'architettura RISC AVR della Atmel. La capacità di eseguire le istruzioni in un singolo ciclo consente a questo dispositivo di raggiungere prestazioni di circa 1 MIPS per MHz, consentendo al progettista di fissare il tradeoff tra consumo di potenza e velocità di calcolo.*
- *Tale dispositivo è dotato di 32 registri general purpose, tutti connessi in modo diretto alla ALU, così da consentire l'accesso simultaneo a due registri e l'esecuzione di istruzioni in un solo ciclo. Le caratteristiche più importanti sono:*

Cosa è un ATMEGA

- *16 MIPS milioni di istruzione per sec*
- *128KB di memoria Flash*
- *4KB di EEPROM*
- *4KB di SRAM*
- *53 linee di I/O*
- *2 USART*
- *1 ADC 8 canali ad approssimazioni successive a 10 bit*
- *Porte di comunicazione Two-wire, Serial Interface e Serial Peripheral Interface*
- *Supporto al protocollo JTAG sia per la programmazione che per l'On Chip Debug*

Proprietà di un Microcontrollore AVR Atmel

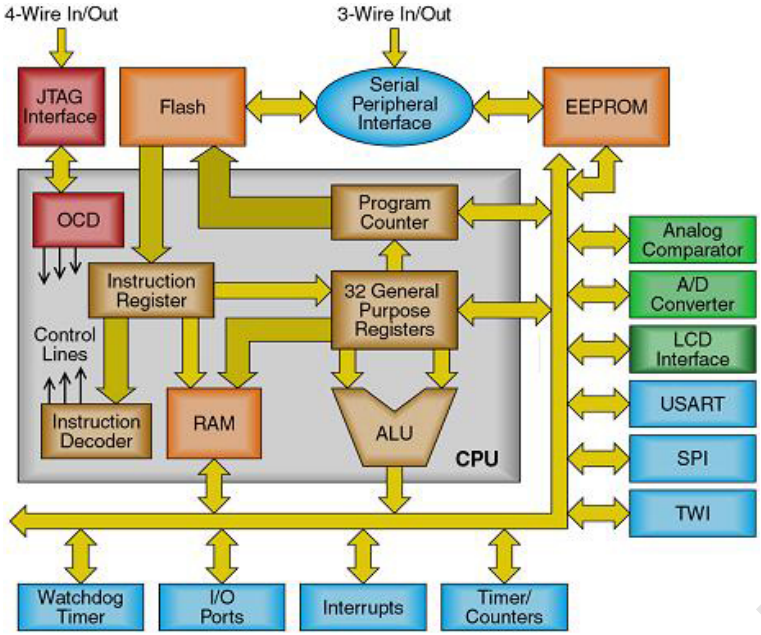
Features

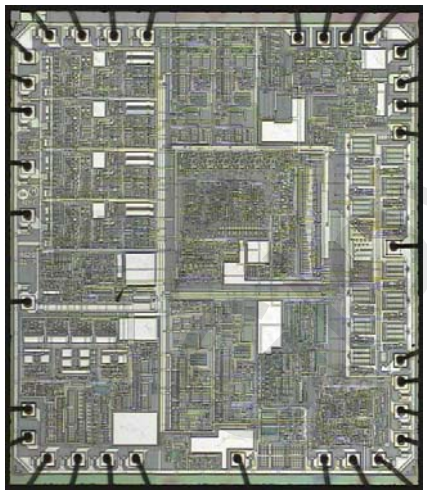
- High-performance, Low-power AVR® 8-bit Microcontroller
- Advanced RISC Architecture
  - 133 Powerful Instructions – Most Single Clock Cycle Execution
  - 32 x 8 General Purpose Working Registers + Peripheral Control Registers
  - Fully Static Operation
  - Up to 16 MIPS Throughput at 16 MHz
  - On-chip 2-cycle Multiplier
- Nonvolatile Program and Data Memories
  - 128K Bytes of In-System Reprogrammable Flash
    Endurance: 1,000 Write/Erase Cycles
  - Optional Boot Code Section with Independent Lock Bits
    In-System Programming by On-chip Boot Program
    True Read-While-Write Operation
  - 4K Bytes EEPROM
    Endurance: 100,000 Write/Erase Cycles
  - 4K Bytes Internal SRAM
  - Up to 64K Bytes Optional External Memory Space
  - Programming Lock for Software Security
  - SPI Interface for In-System Programming
- JTAG (IEEE std. 1149.1 Compliant) Interface
  - Boundary-scan Capabilities According to the JTAG Standard
  - Extensive On-chip Debug Support
  - Programming of Flash, EEPROM, Fuses and Lock Bits through the JTAG Interface

Proprietà di un Microcontrollore AVR Atmel

- Peripheral Features
  - Two 8-bit Timer/Counters with Separate Prescalers and Compare Modes
  - Two Expanded 16-bit Timer/Counters with Separate Prescaler, Compare Mode and Capture Mode
  - Real Time Counter with Separate Oscillator
  - Two 8-bit PWM Channels
  - 6 PWM Channels with Programmable Resolution from 2 to 16 Bits
  - Output Compare Modulator
  - 8-channel, 10-bit ADC
    - 8 Single-ended Channels
    - 7 Differential Channels
    - 2 Differential Channels with Programmable Gain at 1x, 10x, or 200x
  - Byte-oriented Two-wire Serial Interface
  - Dual Programmable Serial USARTs
  - Master/Slave SPI Serial Interface
  - Programmable Watchdog Timer with On-chip Oscillator
  - On-chip Analog Comparator
- Special Microcontroller Features
  - Power-on Reset and Programmable Brown-out Detection
  - Internal Calibrated RC Oscillator
  - External and Internal Interrupt Sources
  - Six Sleep Modes: Idle, ADC Noise Reduction, Power-save, Power-down, Standby, and Extended Standby
  - Software Selectable Clock Frequency
  - ATmega103 Compatibility Mode Selected by a Fuse
  - Global Pull-up Disable
- I/O and Packages
  - 53 Programmable I/O Lines

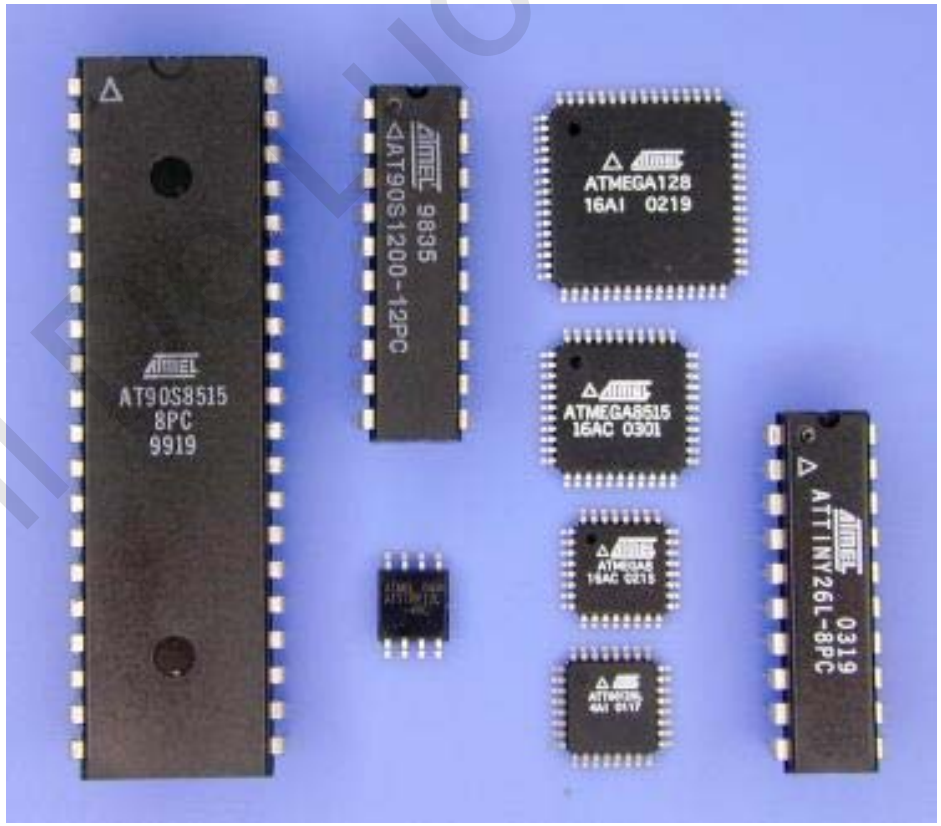
Foto del core di un AVR e vari Packages





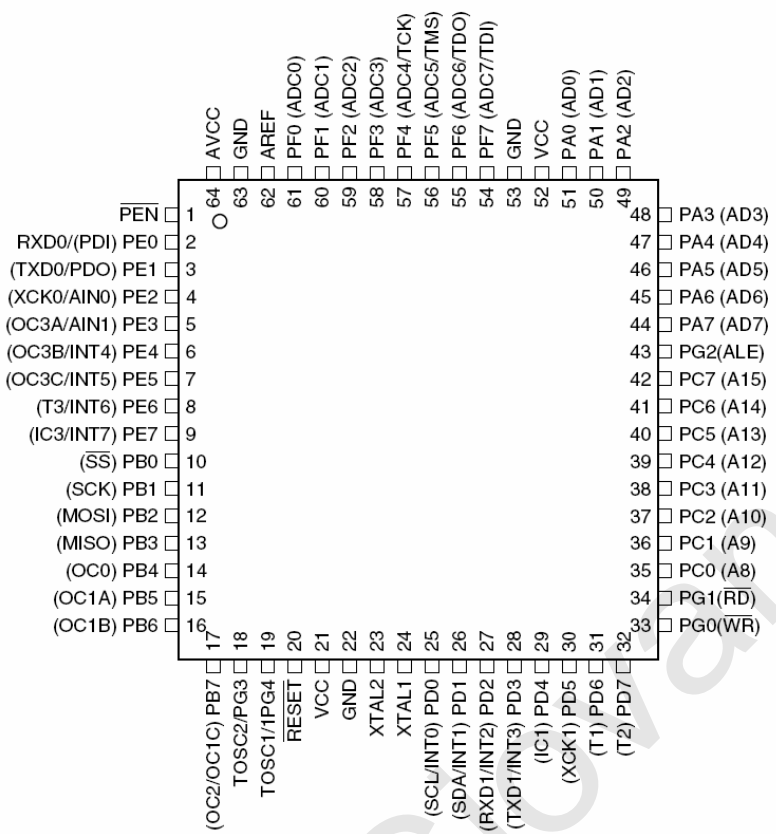
Atmega32

3mm



DIP40, 20, TQFP64, 44, 32

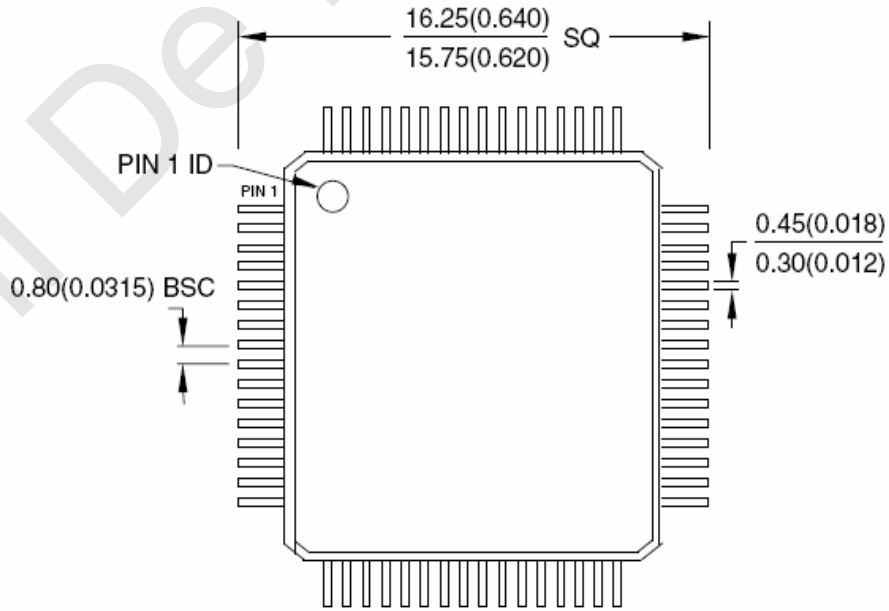
Pinout e Packaging Information ATMEGA128



Packaging Information

64A

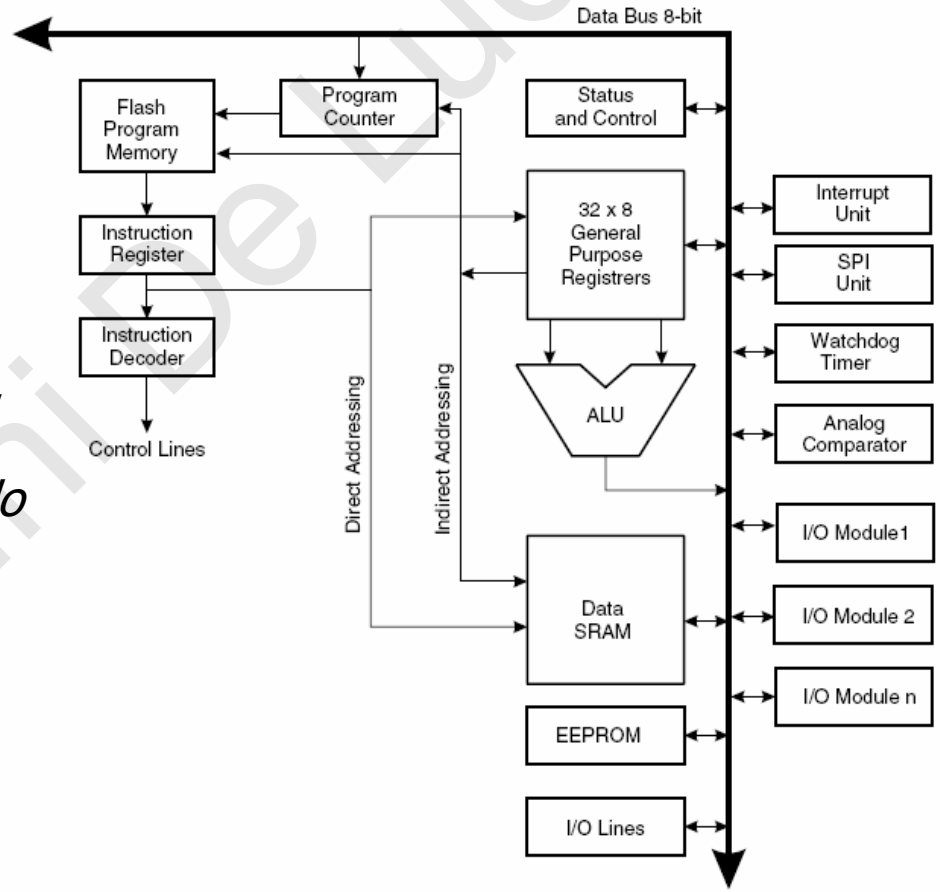
64-lead, Thin (1.0 mm) Plastic Quad Flat Package
(TQFP), 14x14mm body, 2.0mm footprint, 0.8mm pitch.
Dimensions in Millimeters and (Inches)*
JEDEC STANDARD MS-026 AEB



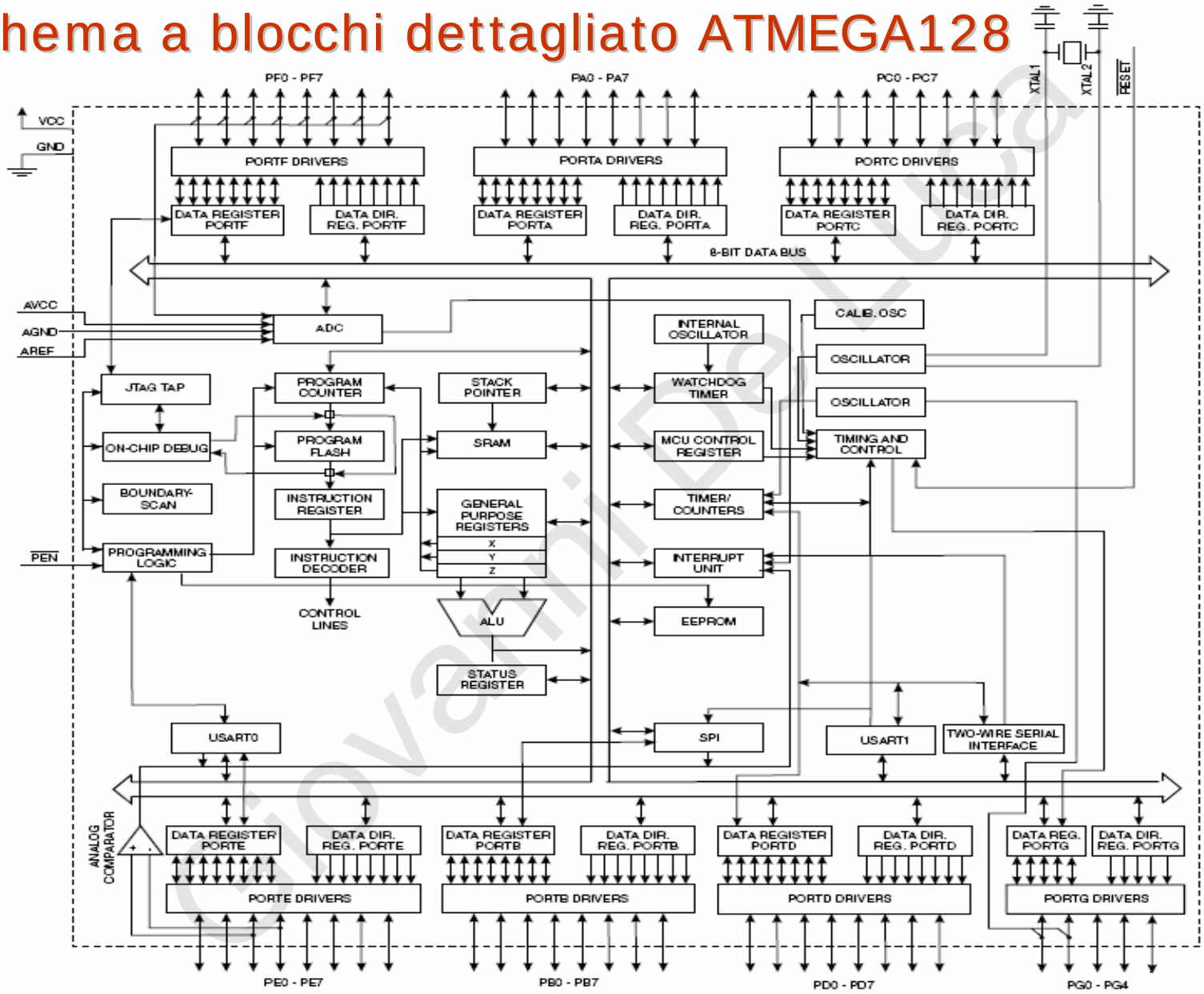
Molti dei pin hanno doppia o tripla funzione a secondo della loro configurazione

Architettura di un ATMEGA128

Per aumentare la performance e il parallelismo, l'AVR usa una architettura Harvard con bus separati per la memoria e per i dati. Le istruzioni nella memoria di programma vengono eseguite con un singolo livello di pipelining. Mentre che una istruzione viene eseguita la prossima istruzione è pre-fecciata dalla memoria programma. Questa modalità permette che le istruzioni vengano eseguite ad ogni ciclo di clock. La ALU supporta operazioni logiche e aritmetiche tra due registri o tra una costante e un registro. Le operazioni su singolo registro possono essere eseguite anche dalla ALU. Dopo una operazione aritmetica, lo Status register è aggiornato e riflette le informazioni circa il risultato della operazione



Schema a blocchi dettagliato ATMEGA128

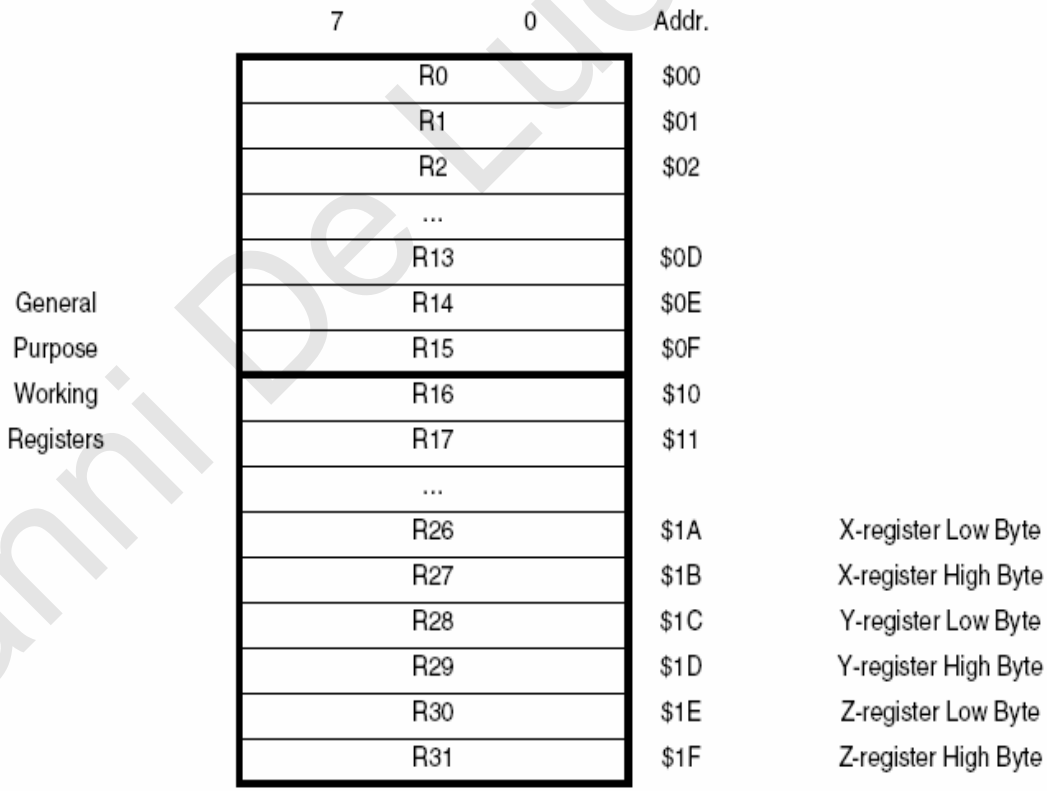


32 Registri Interni general purpose

*Il **'fast-access register file'** contiene 32 x 8 bit registri per uso generico con tempo di accesso di un singolo clock. Questo permette che le operazione della ALU vengano eseguite in un solo ciclo di clock.*

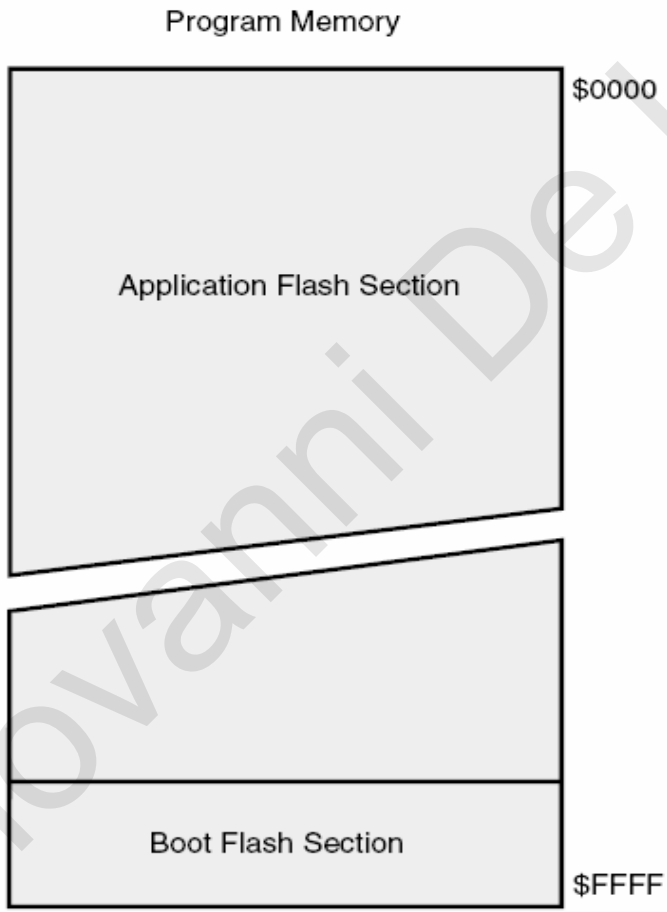
Sei dei 32 registri possono essere usati come tre registri a 16 bit per Data Space Addressing indiretto.

Uno di questi può essere usato per puntare alle tabelle nella Flash program memory.

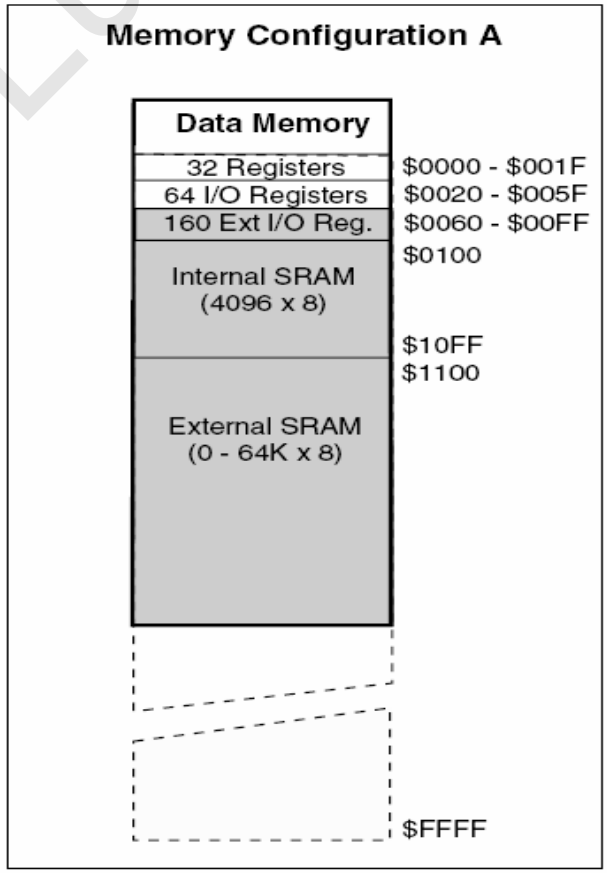


Configurazione della Memoria Interna

L' ATMEGA128 contiene 128Kbyte di Flash Memory riprogrammabile.
Visto che le istruzioni su tutti gli AVR sono a 16 o 32 bit, la Flash è organizzata come 64K x 16bit.
La memoria flash ha la durata di circa 1000 cicli di write/erase.

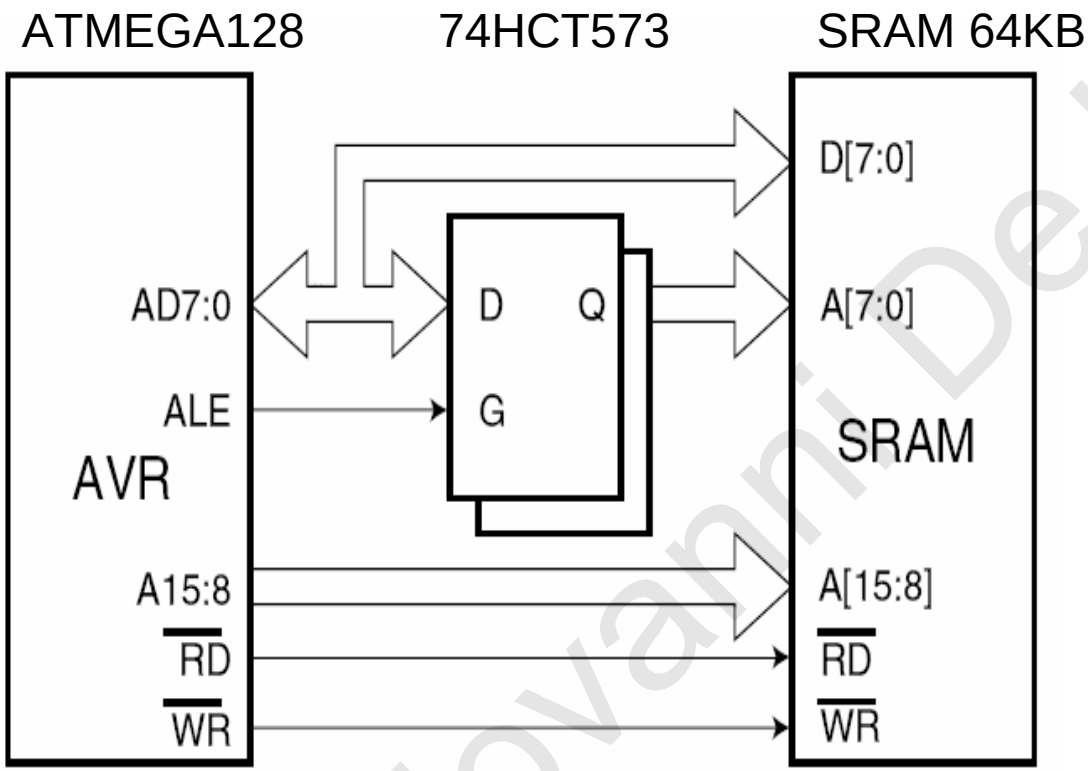


Memoria Programma FLASH



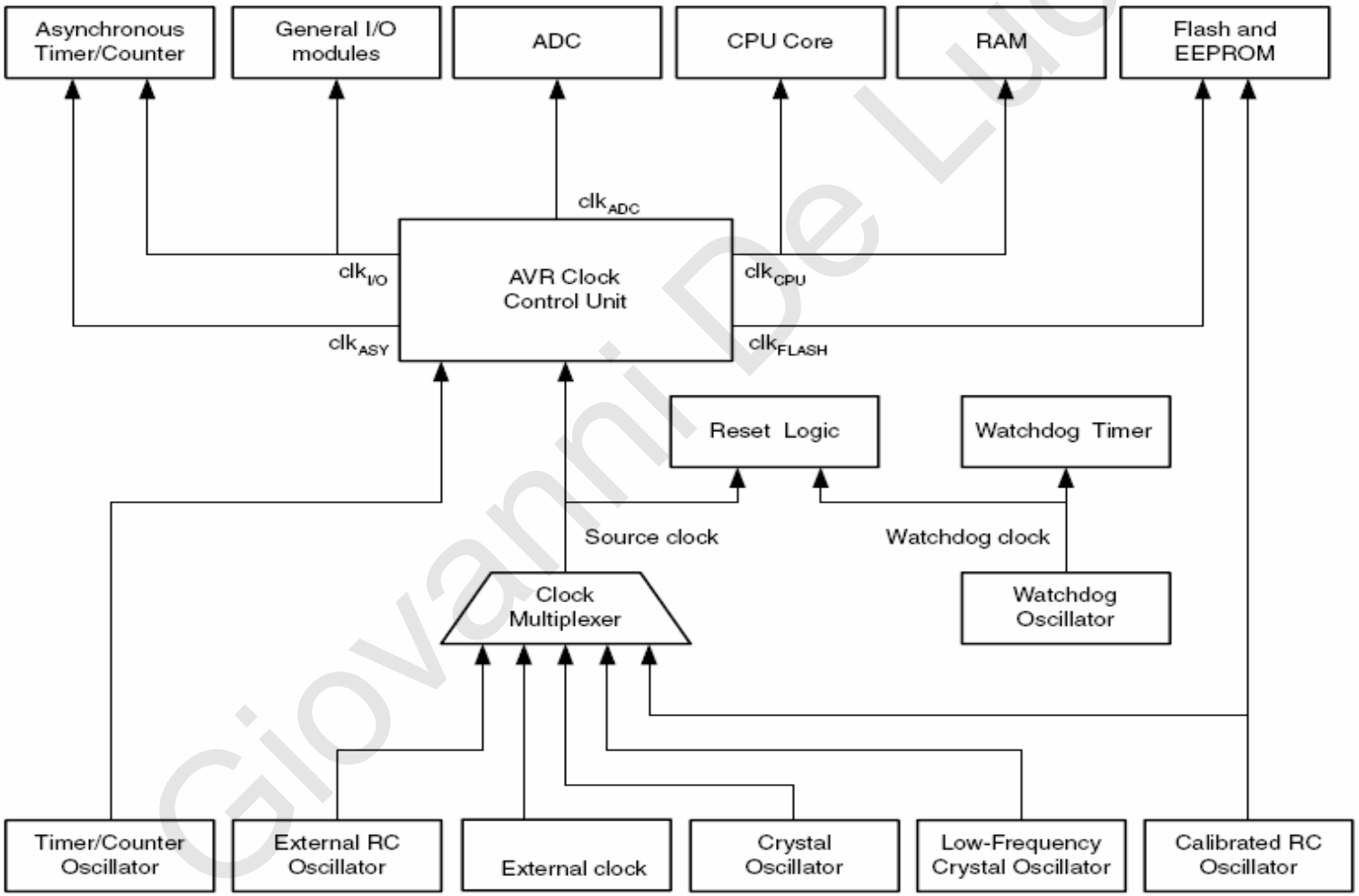
Memoria dati SRAM

Connessione di una SRAM esterna



E' possibile estendere la memoria dati nella maggior parte dei device fino ad un massimo di 64 Kbyte secondo questo schema applicativo.
La stessa configurazione si sfrutta per collegare periferiche esterne mappate in memoria.

Distribuzione del Clock



Segnali di Reset Interni

The ATmega128 ha 5 sorgenti di reset:

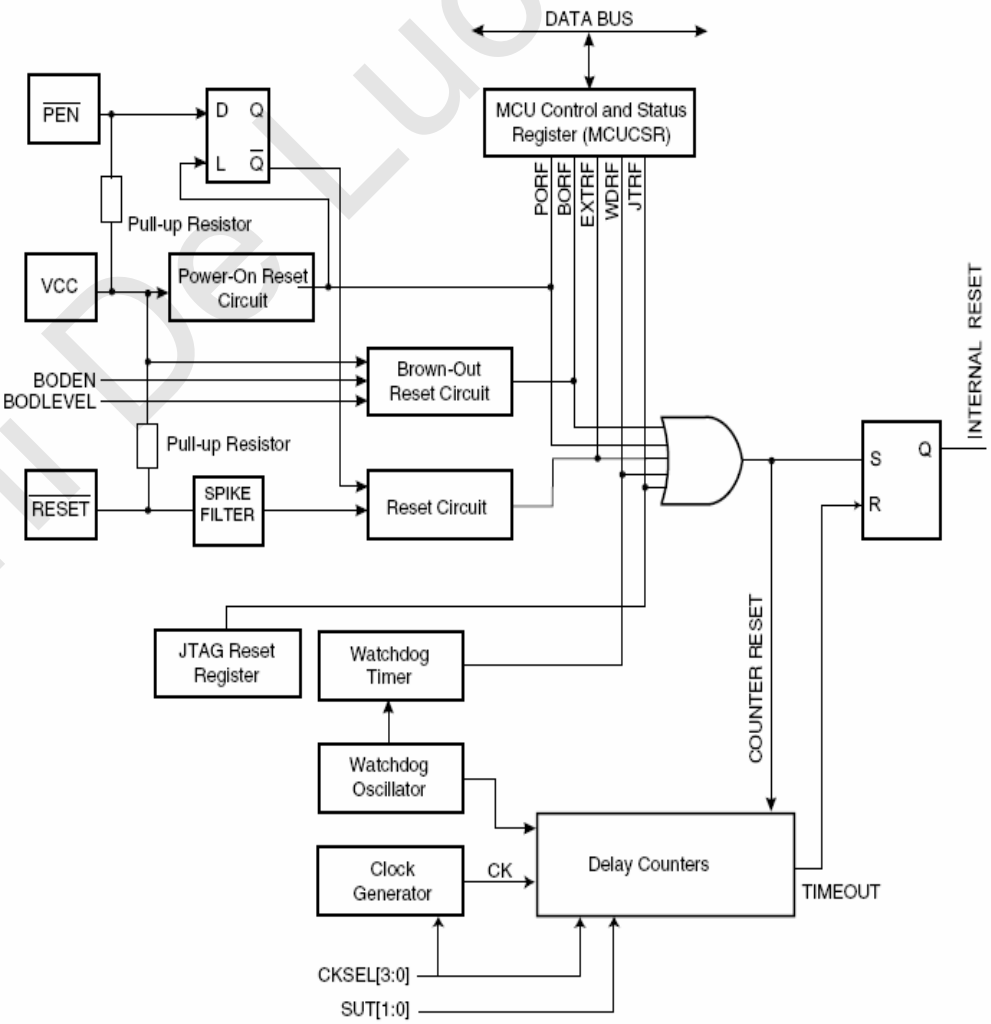
Power-on Reset. *Il MCU è resettato quando la tensione di è sotto il valore di Power-on Reset threshold (VPOT).*

External Reset. *Il MCU è resettato quando un basso livello è presente per una lunga durata sul pin di RESET*

Watchdog Reset. *Il MCU è resettato quando il periodo del Watchdog Timer espira.*

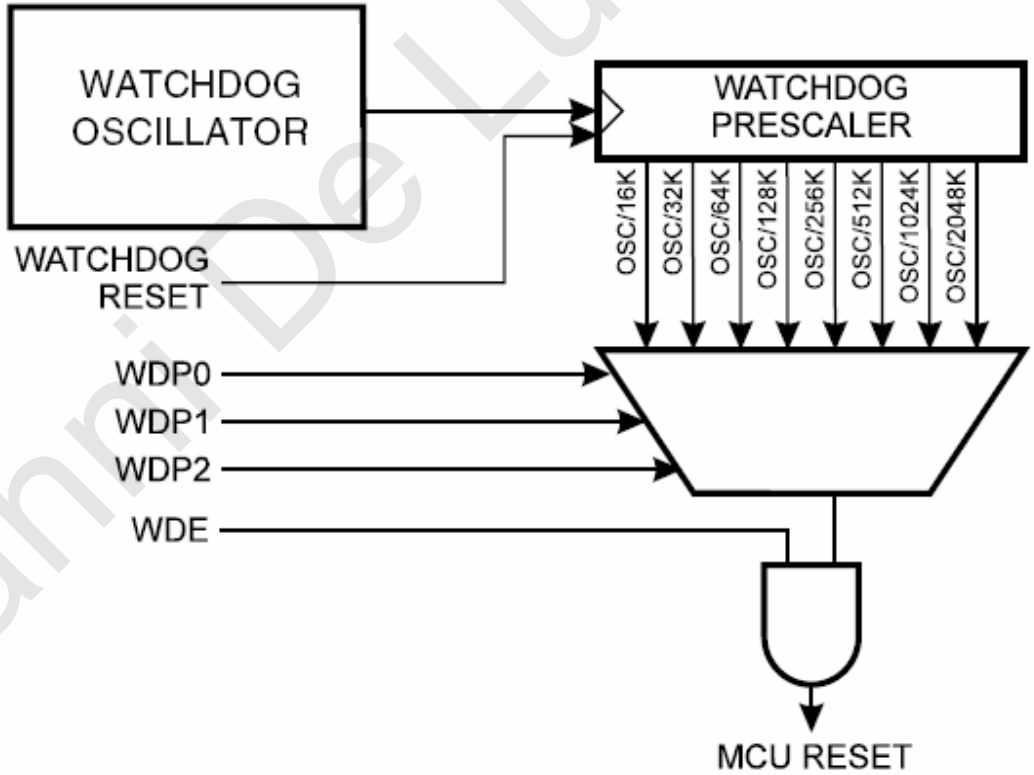
Brown-out Reset. *Il MCU è resettato quando la tensione Vcc è più bassa del valore di Brown-out Reset threshold (VBOT) e il Brown-out Detector e abilitato.*

JTAG AVR Reset. *Il MCU è resettato quando un valore logico 1 è applicato al Reset Register, dello scan chains del sistema JTAG.*



WATCHDOG

Il timer del Watchdog è cloccato con un oscillatore on-chip che lavora a 1 Mhz.
Il WTDC viene resettato dal suo timer, o quando arriva un reset diverso. Possono essere selezionati 8 differenti cicli di clock per determinare il periodo di reset. Se il periodo di reset espira senza un nuovo Watchdog reset, L'ATMEGA128 si resetta e punta al Reset Vector.



Vettori di reset e di interrupts

Vector No.	Program Address[2]	Source	Interrupt Definition
1	$0000[1]	RESET	External Pin, Power-on Reset, Brown-out Reset, Watchdog Reset, and JTAG AVR Reset
2	$0002	INT0	External Interrupt Request 0
3	$0004	INT1	External Interrupt Request 1
4	$0006	INT2	External Interrupt Request 2
5	$0008	INT3	External Interrupt Request 3
6	$000A	INT4	External Interrupt Request 4
7	$000C	INT5	External Interrupt Request 5
8	$000E	INT6	External Interrupt Request 6
9	$0010	INT7	External Interrupt Request 7
10	$0012	TIMER2 COMP	Timer/Counter2 Compare Match
11	$0014	TIMER2 OVF	Timer/Counter2 Overflow
12	$0016	TIMER1 CAPT	Timer/Counter1 Capture Event
13	$0018	TIMER1 COMPA	Timer/Counter1 Compare Match A
14	$001A	TIMER1 COMPB	Timer/Counter1 Compare Match B
15	$001C	TIMER1 OVF	Timer/Counter1 Overflow
16	$001E	TIMER0 COMP	Timer/Counter0 Compare Match
17	$0020	TIMER0 OVF	Timer/Counter0 Overflow
18	$0022	SPI, STC	SPI Serial Transfer Complete
19	$0024	USART0, RX	USART0, Rx Complete
20	$0026	USART0, UDRE	USART0 Data Register Empty
21	$0028	USART0, TX	USART0, Tx Complete
22	$002A	ADC	ADC Conversion Complete
23	$002C	EE READY	EEPROM Ready
24	$002E	ANALOG COMP	Analog Comparator
25	$0030[3]	TIMER1 COMPC	Timer/Countre1 Compare Match C
26	$0032[3]	TIMER3 CAPT	Timer/Counter3 Capture Event
27	$0034[3]	TIMER3 COMPA	Timer/Counter3 Compare Match A
28	$0036[3]	TIMER3 COMPB	Timer/Counter3 Compare Match B
29	$0038[3]	TIMER3 COMPC	Timer/Counter3 Compare Match C
30	$003A[3]	TIMER3 OVF	Timer/Counter3 Overflow

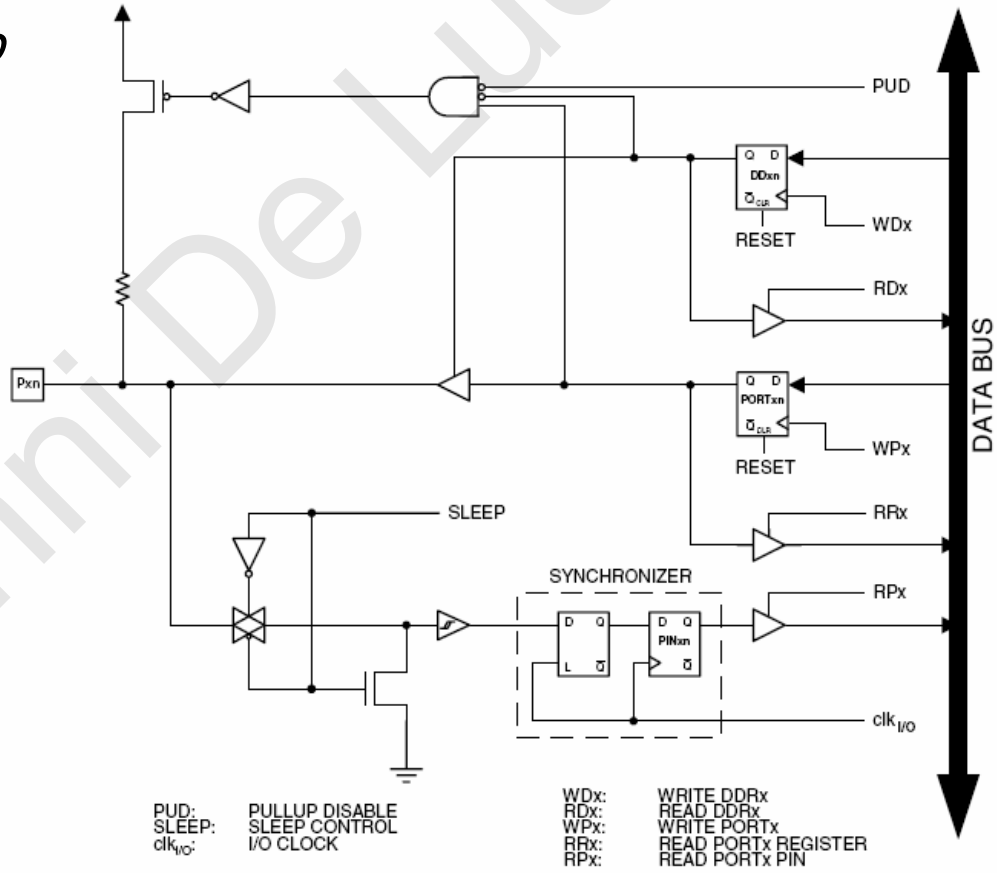
Vector No.	Program Address[2]	Source	Interrupt Definition
31	$003C[3]	USART1, RX	USART1, Rx Complete
32	$003E[3]	USART1, UDRE	USART1 Data Register Empty
33	$0040[3]	USART1, TX	USART1, Tx Complete
34	$0042[3]	TWI	Two-wire Serial Interface
35	$0044[3]	SPM READY	Store Program Memory Ready

Struttura tipica di una linea digitale di I/O

*Tutte le porte degli AVR hanno una funzione di **'true Read-Modify-Write'** quando vengono usate come porte di I/O digitali. Questo significa che le direzioni dei singoli pin può essere cambiata al volo e senza interferire con gli altri pin della stessa porta.*

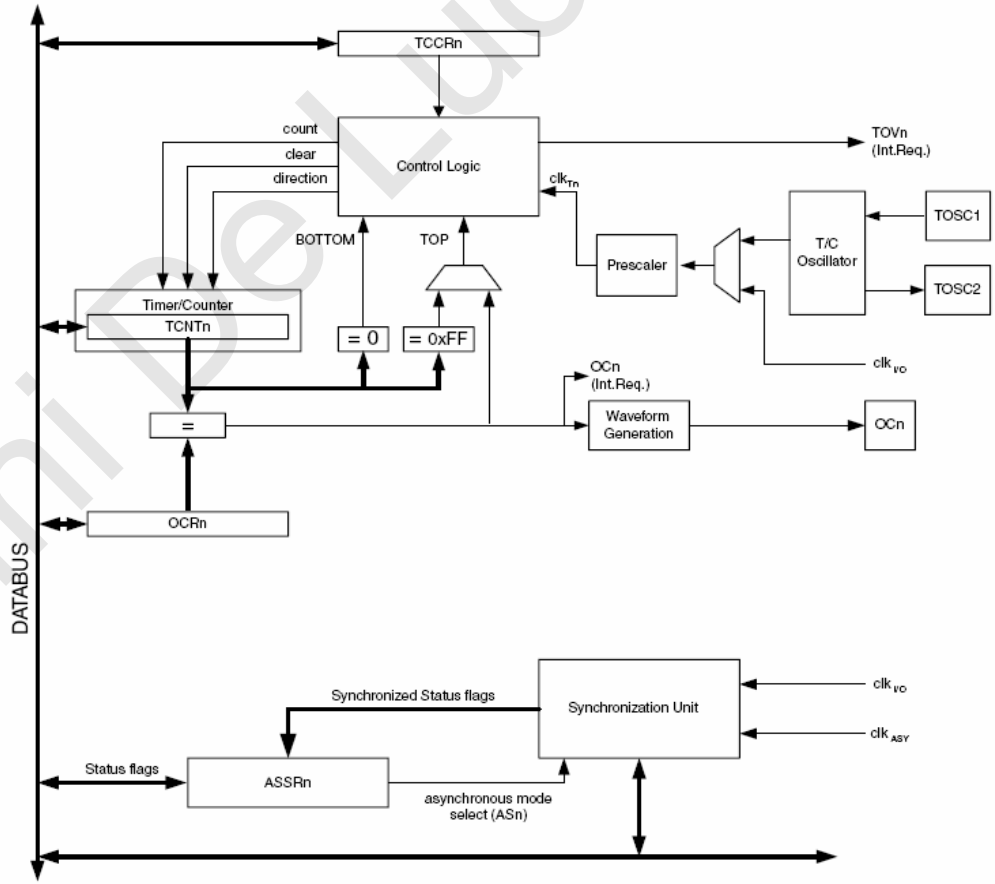
Su ogni singolo pin di input può essere abilitata o disabilitata la resistenza di pull-up. Gli input sono protetti con diodi su Vcc e GND.

I pin di uscita possono direttamente pilotare led e display.



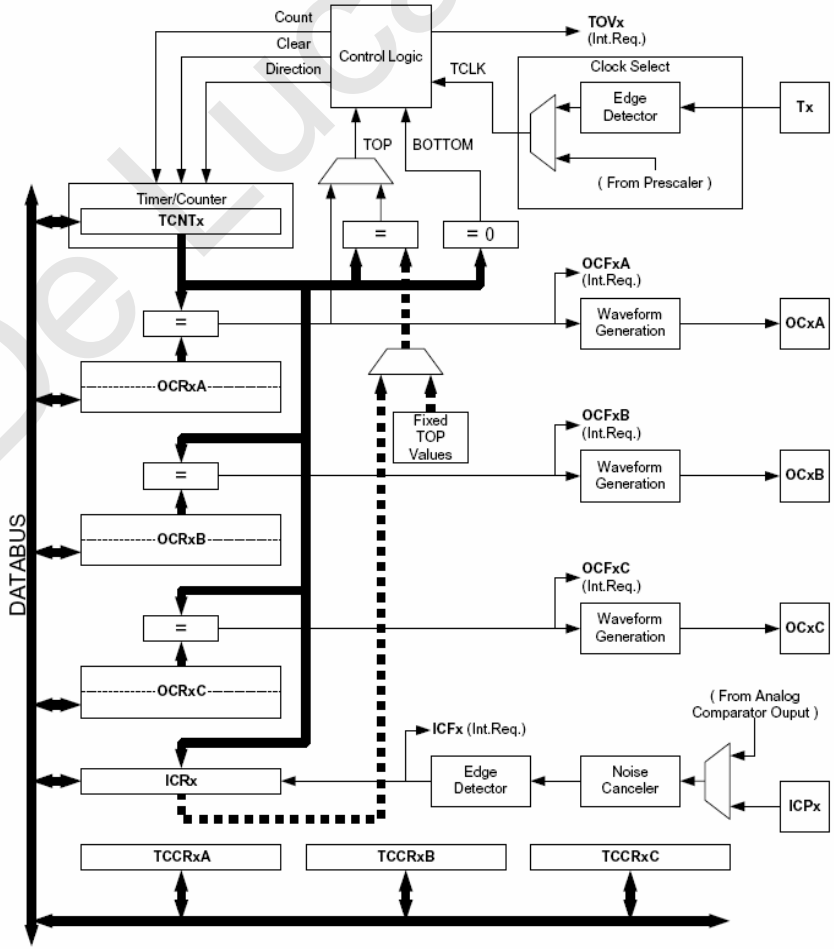
Struttura del Timer 8 Bit

- *Single Channel Counter*
- *Clear Timer on Compare Match (Auto Reload)*
- *Glitch-free, Phase Correct Pulse Width Modulator (PWM)*
- *Frequency Generator*
- *10-bit Clock Prescaler*
- *Overflow and Compare Match Interrupt Sources (TOV0 and OCF0)*
- *Allows Clocking from External 32 kHz Watch Crystal Independent of the I/O Clock*



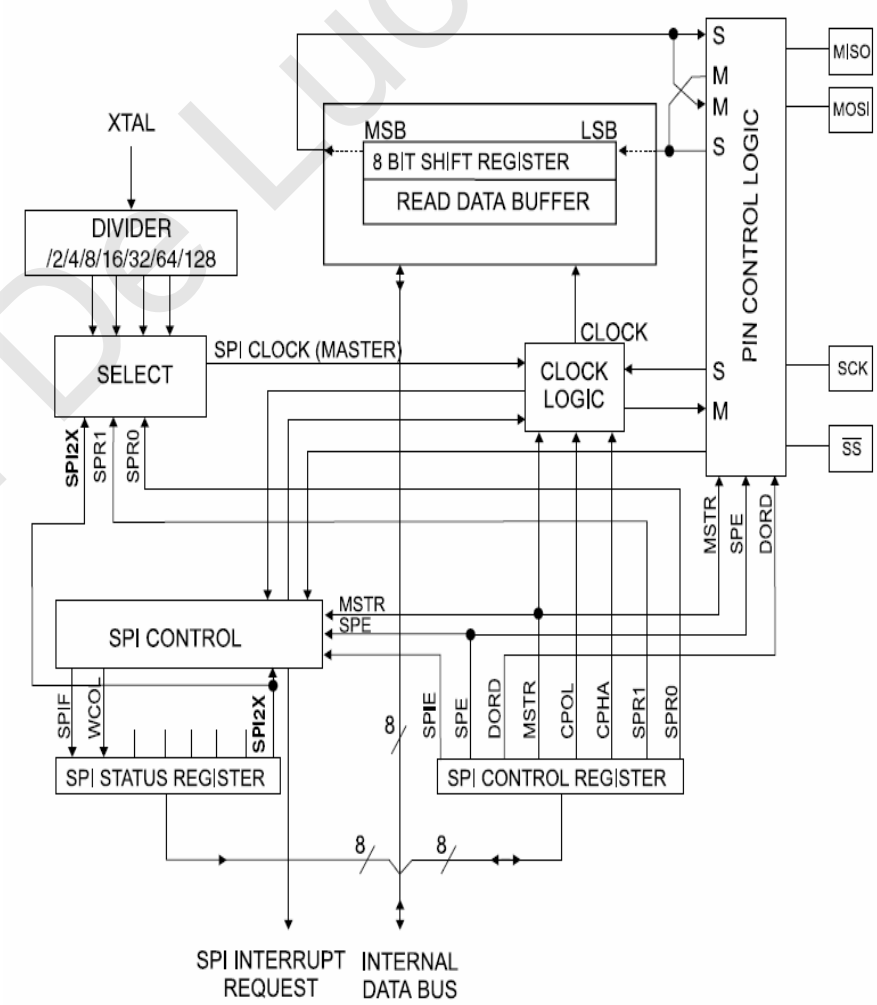
Struttura del Timer 16 bit

- *True 16-bit Design (i.e. ,Allows 16-bit PWM)*
- *Three Independent Output Compare Units*
- *Double Buffered Output Compare Registers*
- *One Input Capture Unit*
- *Input Capture Noise Canceler*
- *Clear Timer on Compare Match (Auto Reload)*
- *Glitch-free, Phase Correct Pulse width Modulator (PWM)*
- *Variable PWM Period*
- *Frequency Generator*
- *External Event Counter*
- *Ten Independent Interrupt Sources (TOV1, OCF1A, OCF1B, OCF1C, ICF1, TOV3, OCF3A, OCF3B, OCF3C, and ICF3)*

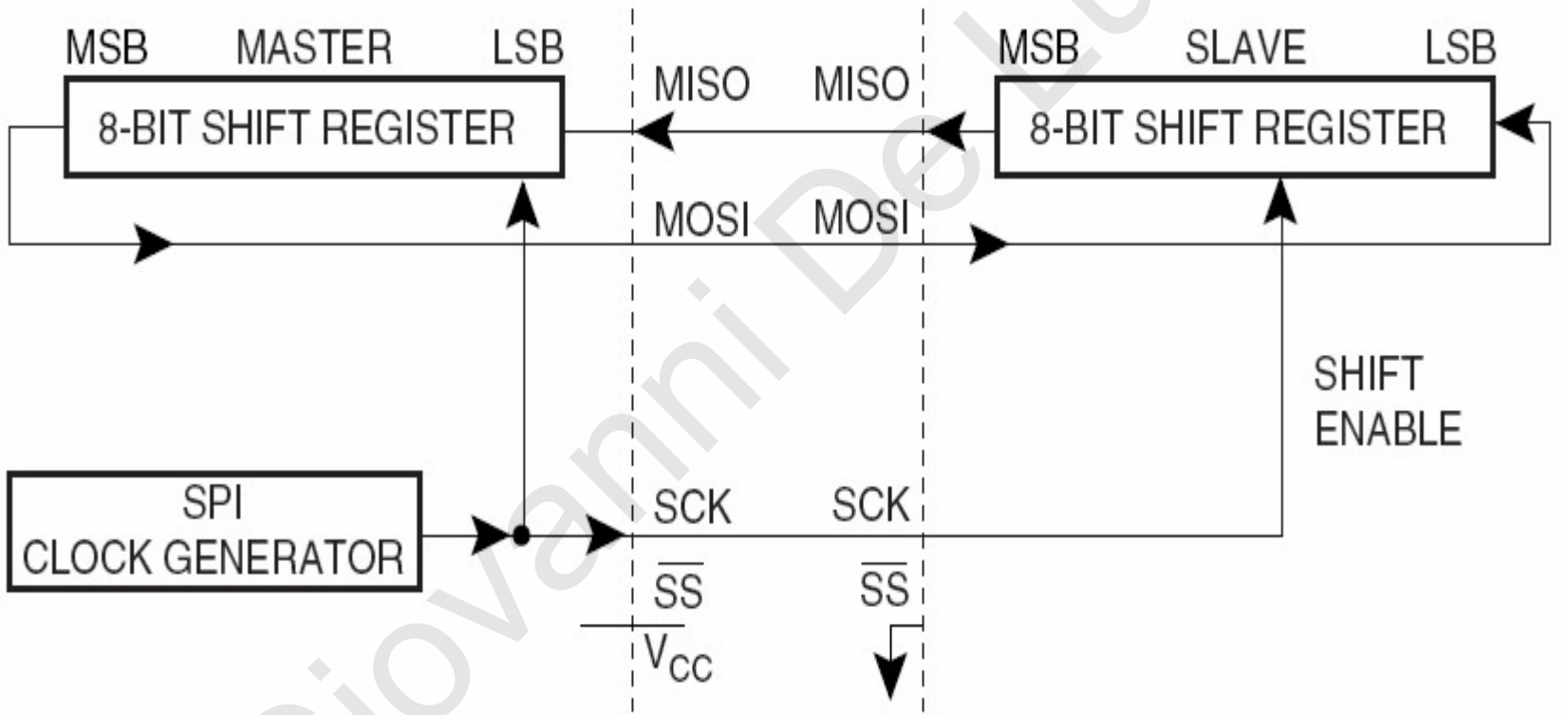


Schema a blocchi della SPI

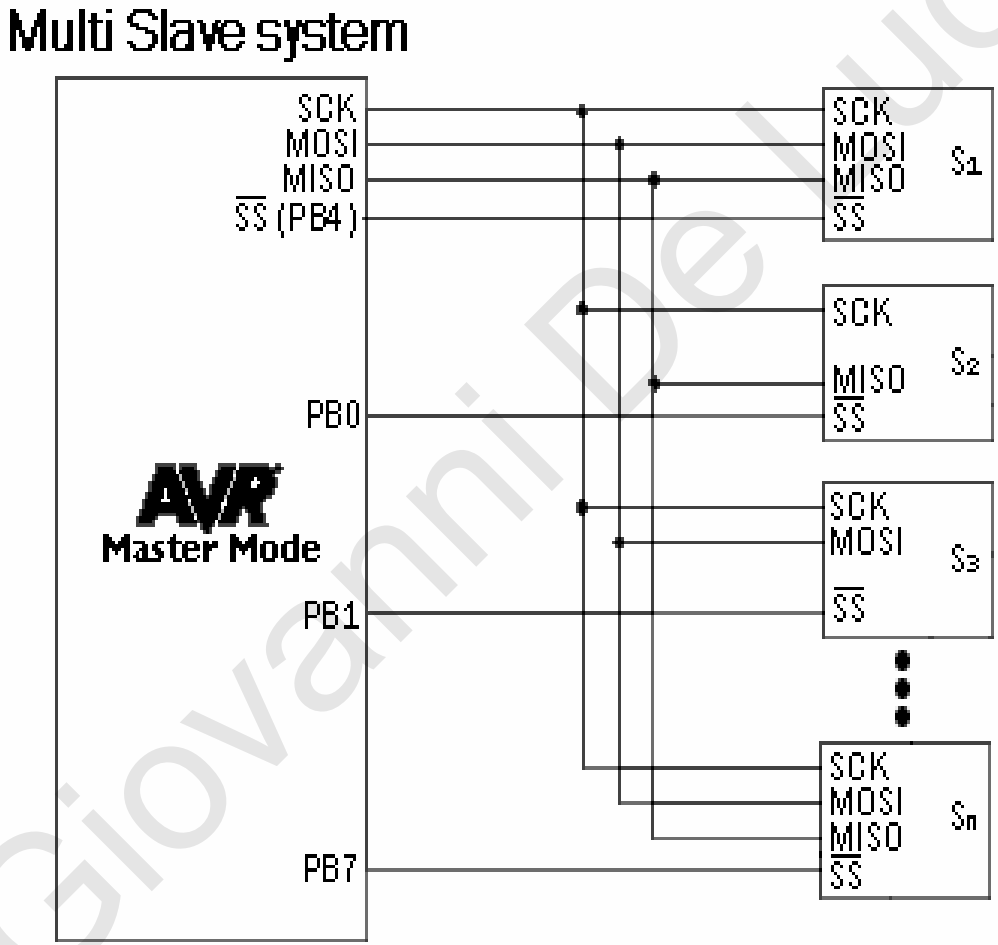
- *Full-duplex, Three-wire Synchronous Data Transfer*
- *Master or Slave Operation*
- *LSB First or MSB First Data Transfer*
- *Seven Programmable Bit Rates*
- *End of Transmission Interrupt Flag*
- *Write Collision Flag Protection*
- *Wake-up from Idle Mode*
- *Double Speed (CK/2) Master SPI Mode*



SPI Master-Slave Interconnection



Sistema di interconnessione SPI multimaster



SPI Transfer Format con CPHA=0

Figure 62. SPI Transfer Format with CPHA = 0

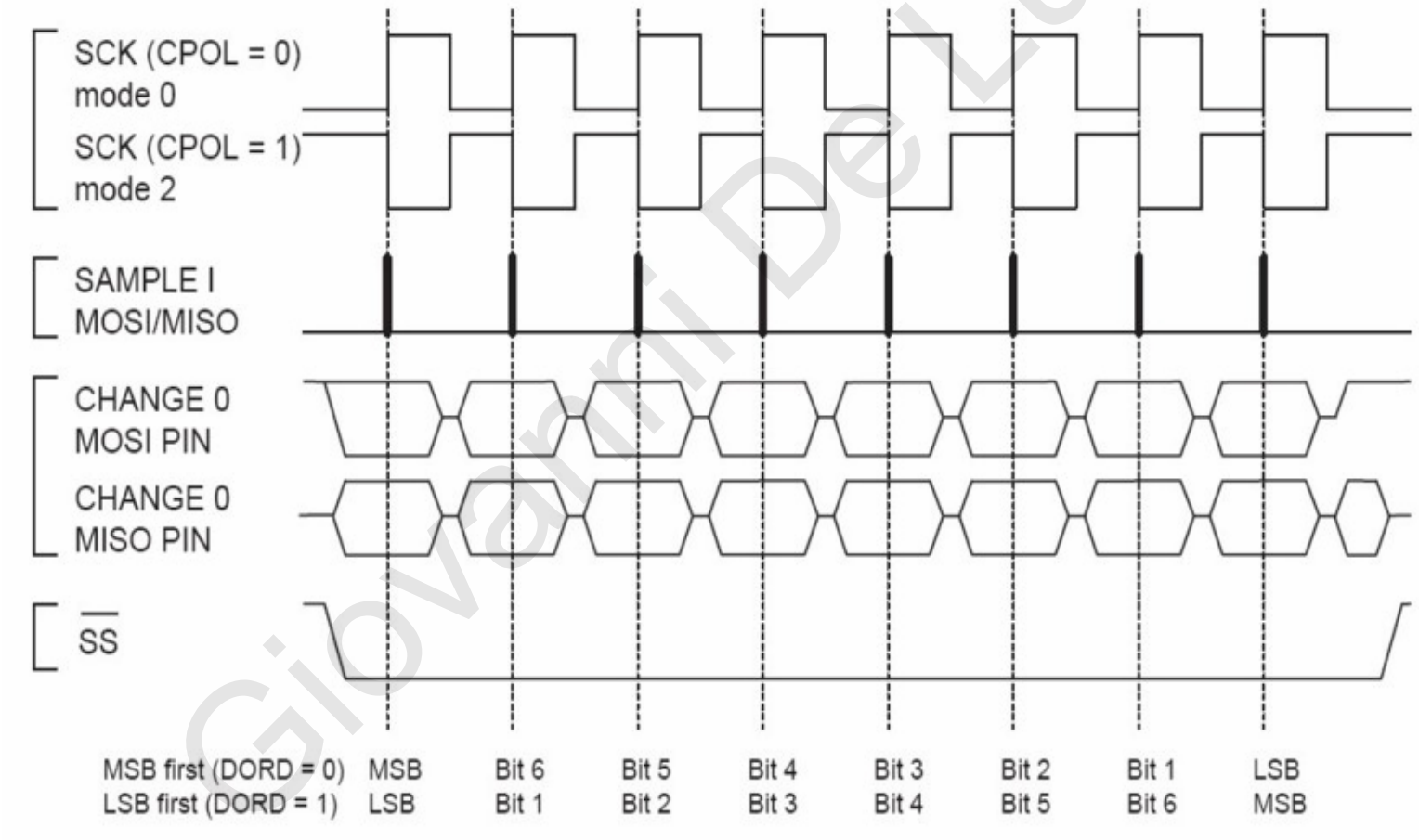
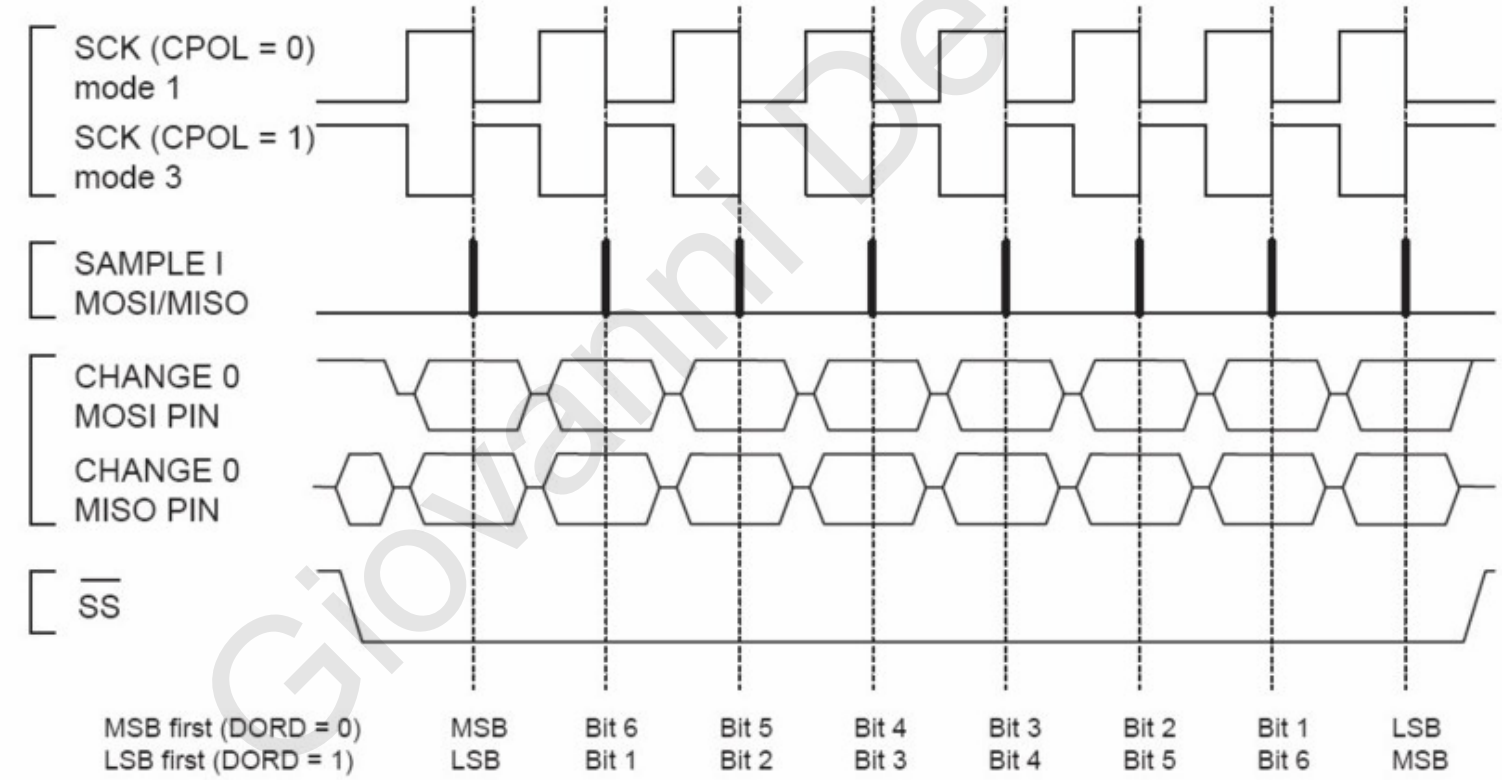


Figure 63. SPI Transfer Format with CPHA = 1



SPI Pin e Uso di SS

Table 1. SPI Pin Overrides

Pin Direction Overrides	Master SPI Mode Direction Overrides	Slave SPI Modes
MOSI	User Defined	Input
MISO	Input	User Defined
SCK	User Defined	Input
SS	User Defined	Input

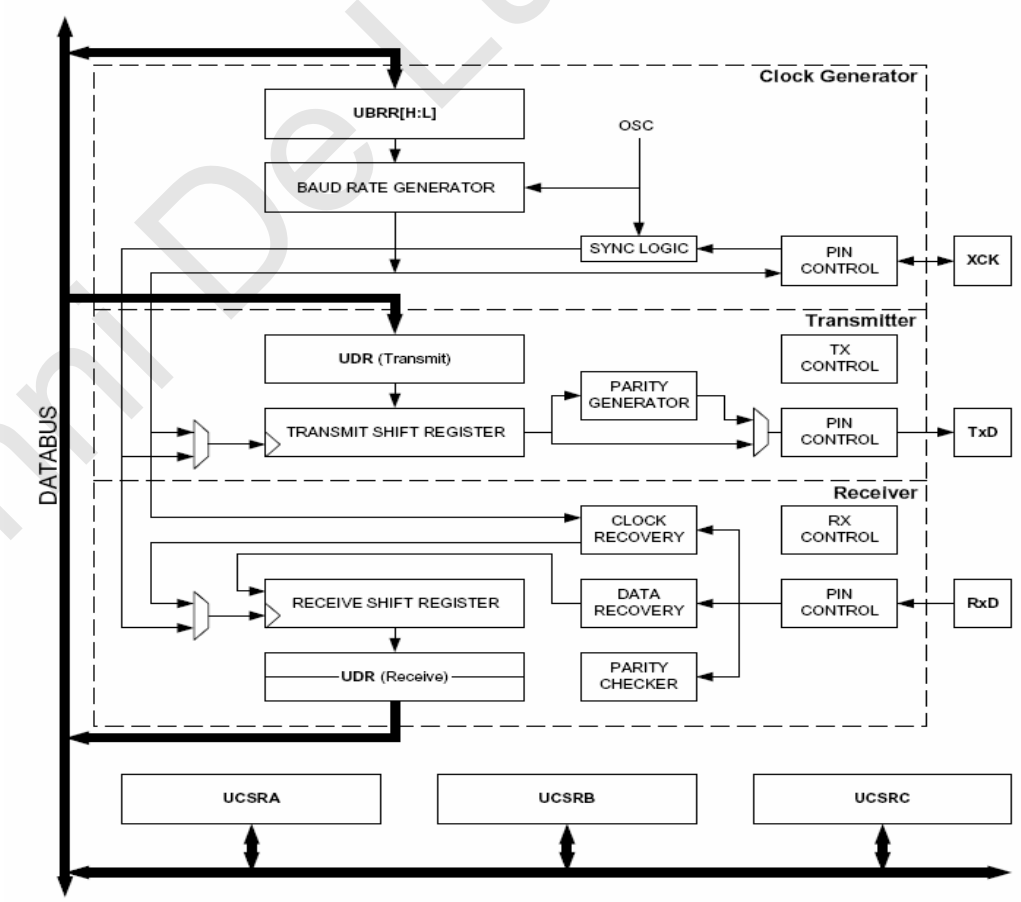
TABLE 2. Overview of SS pin.

Mode	**/SS Config**	**/SS Pin level**	**Description**
Slave	Always input	High	Slave deactivated
		Low	Slave activated
Master	Input	High	Master activated
		Low	Master deactivated
	Output	High	Master activated
		Low	

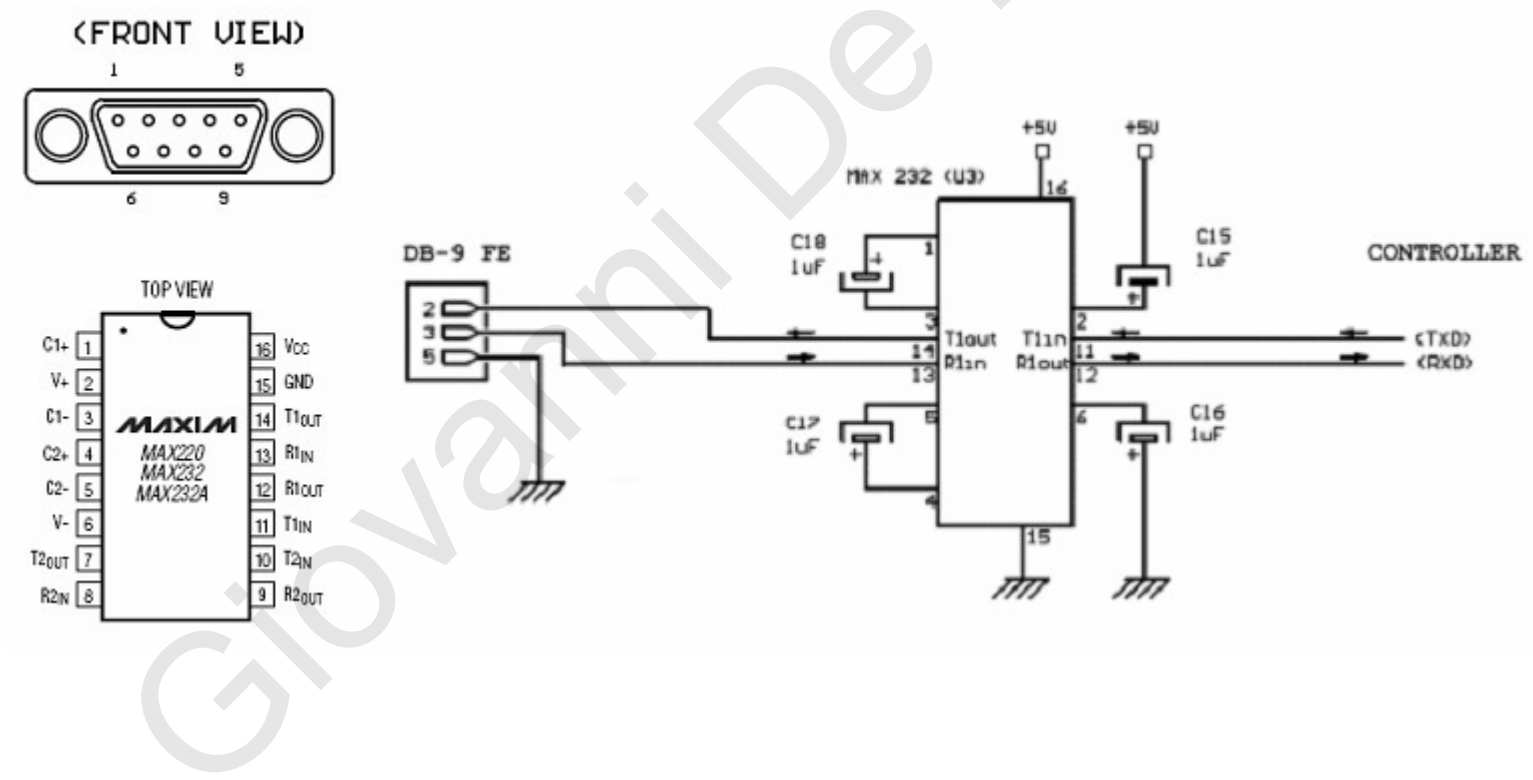
Schema a blocchi della UART

- *Full Duplex Operation (Independent Serial Receive and Transmit Registers)*
- *Asynchronous or Synchronous Operation*
- *Master or Slave Clocked Synchronous Operation*
- *High Resolution Baud Rate Generator*
- *Supports Serial Frames with 5, 6, 7, 8, or 9 Data Bits and 1 or 2 Stop Bits*
- *Odd or Even Parity Generation and Parity Check Supported by Hardware*
- *Data OverRun Detection*
- *Framing Error Detection*
- *Noise Filtering Includes False Start Bit Detection and Digital Low Pass Filter*
- *Three Separate Interrupts on TX Complete, TX Data Register Empty, and RX Complete*
- *Multi-processor Communication Mode*
- *Double Speed Asynchronous Communication Mode*

Universal Asynchronous Receiver Transmitter



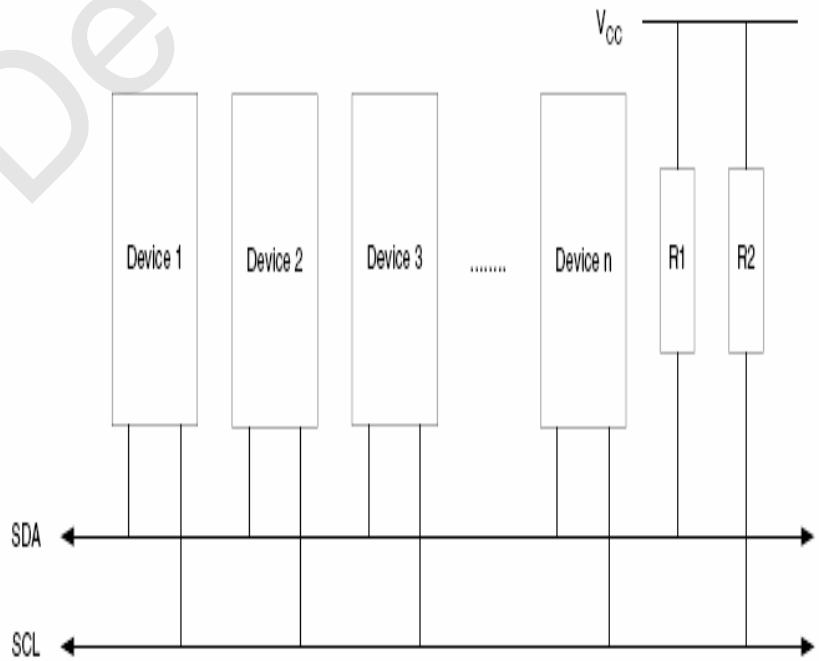
UART Hardware Connection



Interconnessione BUS TWI Two wire interface

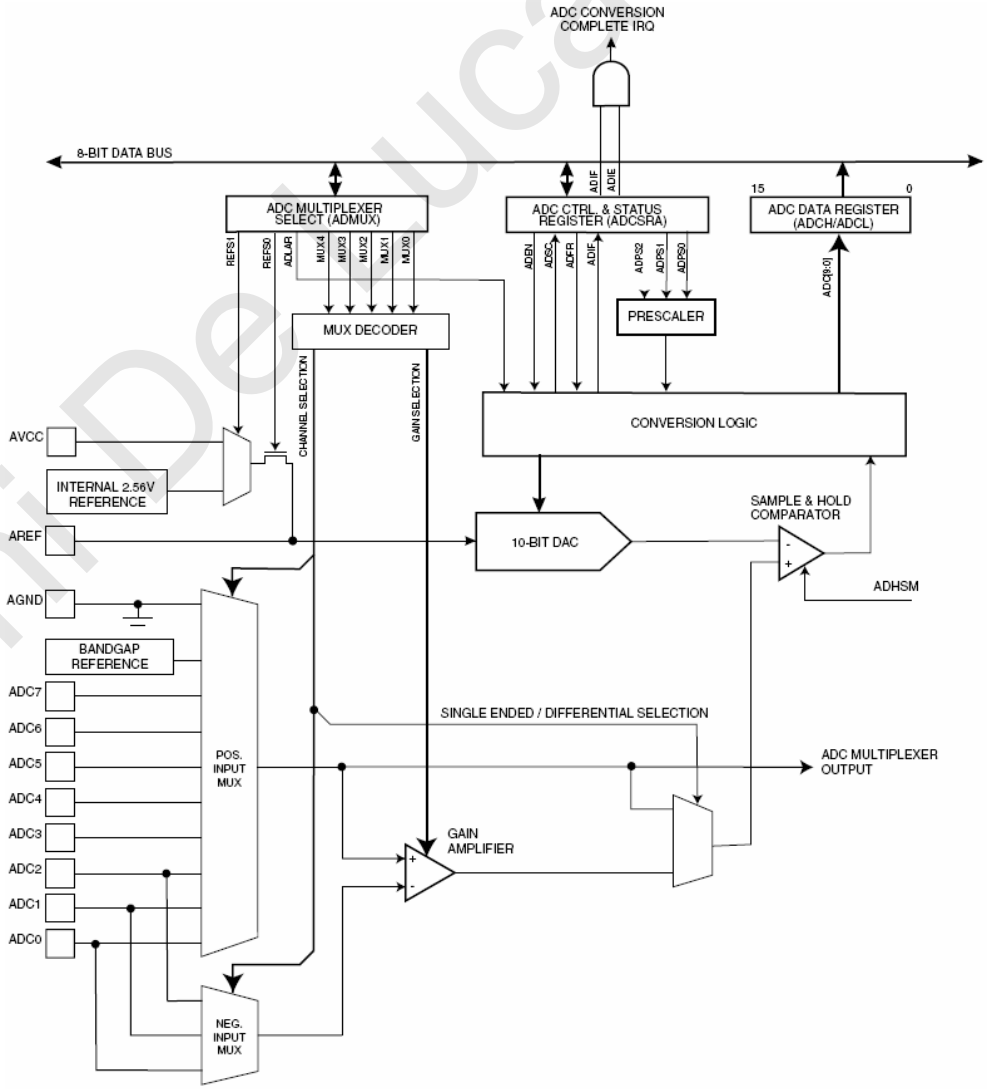
Internal-Integrated Circuits Bus I2C Philips

- *Simple yet Powerful and Flexible Communication Interface, only Two Bus Lines Needed*
- *Both Master and Slave Operation Supported*
- *Device can Operate as Transmitter or Receiver*
- *7-bit Address Space allows up to 128 Different Slave Addresses*
- *Multi-master Arbitration Support*
- *Up to 400 kHz Data Transfer Speed*
- *Slew-rate Limited Output Drivers*
- *Noise Suppression Circuitry Rejects Spikes on Bus Lines*
- *Fully Programmable Slave Address with General Call Support*
- *Address Recognition Causes Wake-up when AVR is in Sleep Mode*

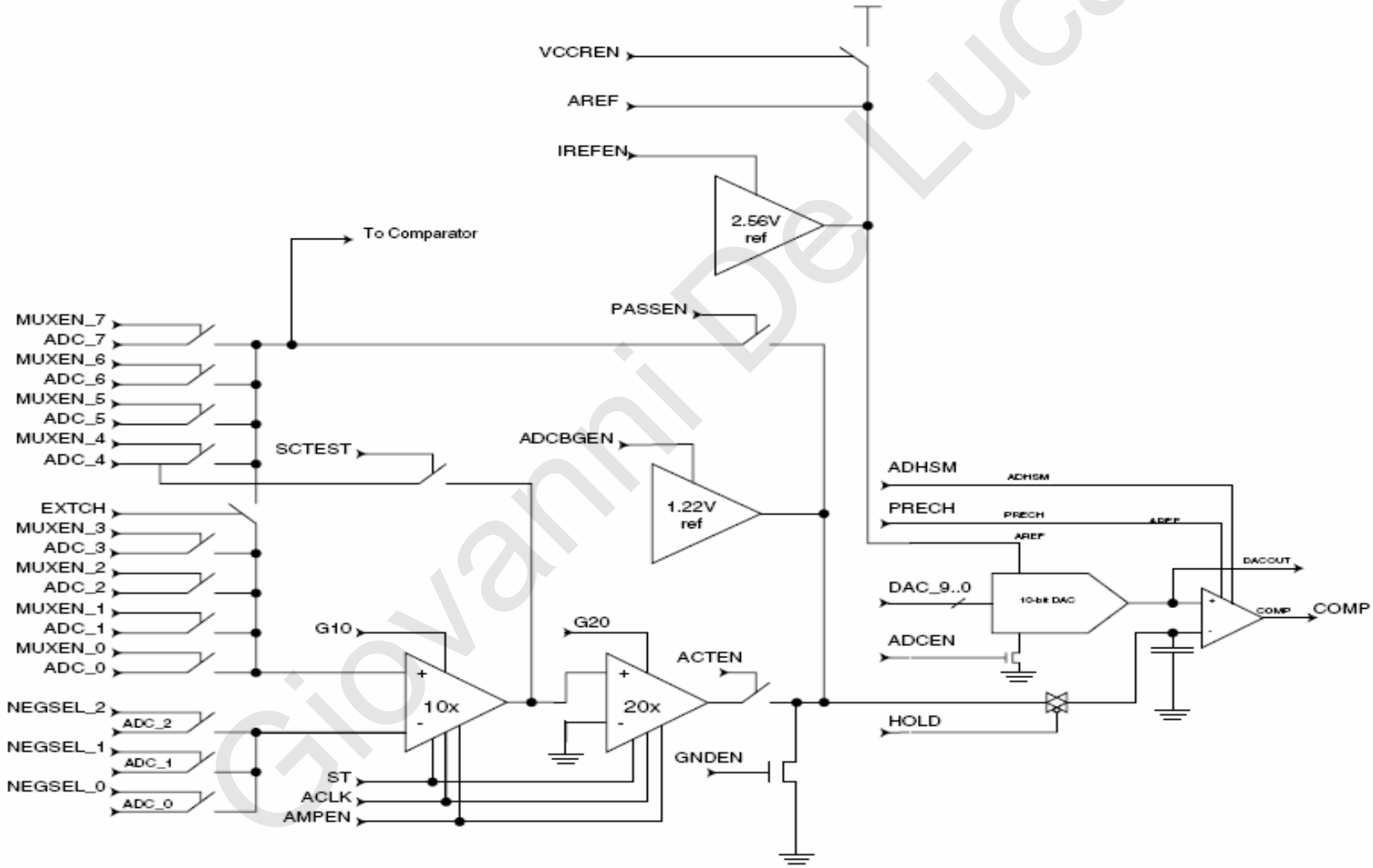


Schema a blocchi ADC interno

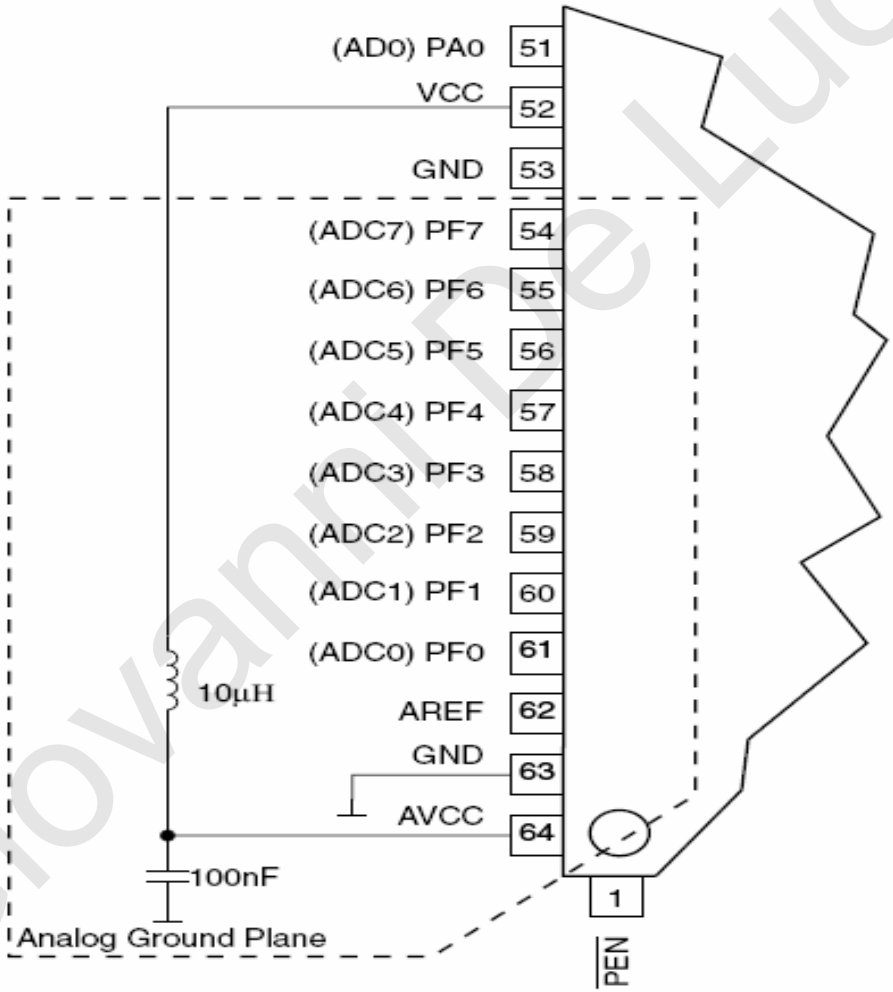
- *10-bit Resolution*
- *0.5 LSB Integral Non-linearity*
- *±2 LSB Absolute Accuracy*
- *65 - 260 µs Conversion Time*
- *Up to 15 kSPS at Maximum Resolution*
- *8 Multiplexed Single Ended Input Channels*
- *7 Differential Input Channels*
- *2 Differential Input Channels with Optional*
  Gain of 10x and 200x
- *Optional Left Adjustment for ADC Result Readout*
- *0 - VCC ADC Input Voltage Range*
- *Selectable 2.56 V ADC Reference Voltage*
- *Free Running or Single Conversion Mode*
- *Interrupt on ADC Conversion Complete*
- *Sleep Mode Noise Canceler*



Struttura interna ADC

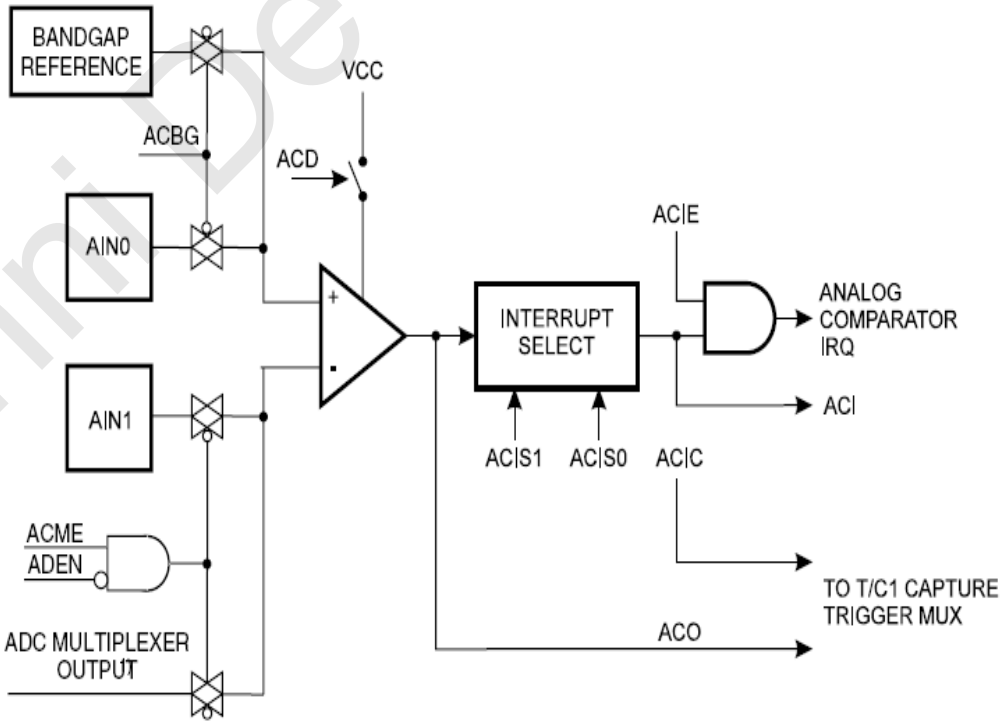


Come alimentare l'ADC ATMEGA128



Struttura Analog Comparator

L' ***Analog Comparator*** *compara il valore positivo presentato sul pin AINO con quello negativo presentato sul pin AIN1. Quando la tensione sul pin positivo è più alta di quello negativo l'Analog Comparator Output, ACO, viene settato.*



Il set di istruzioni Assembly dei micro AVR

Arithmetic and Logic Instructions

Mnemonic	Operands	Description	Operation	Flags	Cycles
ADD	Rd,Rr	Add without Carry	Rd = Rd + Rr	Z,C,N,V,H,S	1
ADC	Rd,Rr	Add with Carry	Rd = Rd + Rr + C	Z,C,N,V,H,S	1
SUB	Rd,Rr	Subtract without Carry	Rd = Rd - Rr	Z,C,N,V,H,S	1
SUBI	Rd,K8	Subtract Immediate	Rd = Rd - K8	Z,C,N,V,H,S	1
SBC	Rd,Rr	Subtract with Carry	Rd = Rd - Rr - C	Z,C,N,V,H,S	1
SBCI	Rd,K8	Subtract with Carry Immedtiate	Rd = Rd - K8 - C	Z,C,N,V,H,S	1
AND	Rd,Rr	Logical AND	Rd = Rd · Rr	Z,N,V,S	1
ANDI	Rd,K8	Logical AND with Immediate	Rd = Rd · K8	Z,N,V,S	1
OR	Rd,Rr	Logical OR	Rd = Rd V Rr	Z,N,V,S	1
ORI	Rd,K8	Logical OR with Immediate	Rd = Rd V K8	Z,N,V,S	1
EOR	Rd,Rr	Logical Exclusive OR	Rd = Rd EOR Rr	Z,N,V,S	1
COM	Rd	One's Complement	Rd = $FF - Rd	Z,C,N,V,S	1
NEG	Rd	Two's Complement	Rd = $00 - Rd	Z,C,N,V,H,S	1

I flags degli AVR

- *Z – zero*
- *C – carry*
- *N – negative*
- *V – overflow*
- *H – half-carry*
- *S – sign bit*
- *I – interrupt enable bit*
- *T – transfer bit*

Arithmetic and Logic Instructions

SBR	Rd,K8	Set Bit(s) in Register	Rd = Rd V K8	Z,C,N,V,S	1
CBR	Rd,K8	Clear Bit(s) in Register	Rd = Rd · ($FF - K8)	Z,C,N,V,S	1
INC	Rd	Increment Register	Rd = Rd + 1	Z,N,V,S	1
DEC	Rd	Decrement Register	Rd = Rd -1	Z,N,V,S	1
TST	Rd	Test for Zero or Negative	Rd = Rd · Rd	Z,C,N,V,S	1
CLR	Rd	Clear Register	Rd = 0	Z,C,N,V,S	1
SER	Rd	Set Register	Rd = $FF	None	1
ADIW	Rdl,K6	Add Immediate to Word	Rdh:Rdl = Rdh:Rdl + K6	Z,C,N,V,S	2
SBIW	Rdl,K6	Subtract Immediate from Word	Rdh:Rdl = Rdh:Rdl - K 6	Z,C,N,V,S	2
MUL	Rd,Rr	Multiply Unsigned	R1:R0 = Rd * Rr	Z,C	2
MULS	Rd,Rr	Multiply Signed	R1:R0 = Rd * Rr	Z,C	2
MULSU	Rd,Rr	Multiply Signed with Unsigned	R1:R0 = Rd * Rr	Z,C	2
FMUL	Rd,Rr	Fractional Multiply Unsigned	R1:R0 = (Rd * Rr) << 1	Z,C	2
FMULS	Rd,Rr	Fractional Multiply Signed	R1:R0 = (Rd *Rr) << 1	Z,C	2
FMULSU	Rd,Rr	Fractional Multiply Signed with Unsigned	R1:R0 = (Rd * Rr) << 1	Z,C	2

Branch Instructions

Mnemonic	Operands	Description	Operation	Flags	Cycles
RJMP	k	Relative Jump	PC = PC + k +1	None	2
IJMP	None	Indirect Jump to (Z)	PC = Z	None	2
EIJMP	None	Extended Indirect Jump (Z)	STACK = PC+1, PC(15:0) = Z, PC(21:16) = EIND	None	2
JMP	k	Jump	PC = k	None	3
RCALL	k	Relative Call Subroutine	STACK = PC+1, PC = PC + k + 1	None	3/4*
ICALL	None	Indirect Call to (Z)	STACK = PC+1, PC = Z	None	3/4*
EICALL	None	Extended Indirect Call to (Z)	STACK = PC+1, PC(15:0) = Z, PC(21:16) =EIND	None	4*
CALL	k	Call Subroutine	STACK = PC+2, PC = k	None	4/5*
RET	None	Subroutine Return	PC = STACK	None	4/5*
RETI	None	Interrupt Return	PC = STACK	I	4/5*
CPSE	Rd,Rr	Compare, Skip if equal	if (Rd ==Rr) PC = PC 2 or 3	None	1/2/3
CP	Rd,Rr	Compare	Rd -Rr	Z,C,N,V,H,S	1
CPC	Rd,Rr	Compare with Carry	Rd - Rr - C	Z,C,N,V,H,S	1
CPI	Rd,K8	Compare with Immediate	Rd - K	Z,C,N,V,H,S	1
SBRC	Rr,b	Skip if bit in register cleared	if(Rr(b)==0) PC = PC + 2 or 3	None	1/2/3

Branch Instructions

Mnemonic	Operands	Description	Operation	Flags	Cycles
SBRS	Rr,b	Skip if bit in register set	if(Rr(b)==1) PC = PC + 2 or 3	None	1/2/3
SBIC	P,b	Skip if bit in I/O register cleared	if(I/O(P,b)==0) PC = PC + 2 or 3	None	1/2/3
SBIS	P,b	Skip if bit in I/O register set	if(I/O(P,b)==1) PC = PC + 2 or 3	None	1/2/3
BRBC	s,k	Branch if Status flag cleared	if(SREG(s)==0) PC = PC + k + 1	None	1/2
BRBS	s,k	Branch if Status flag set	if(SREG(s)==1) PC = PC + k + 1	None	1/2
BREQ	k	Branch if equal	if(Z==1) PC = PC + k + 1	None	1/2
BRNE	k	Branch if not equal	if(Z==0) PC = PC + k + 1	None	1/2
BRCS	k	Branch if carry set	if(C==1) PC = PC + k + 1	None	1/2
BRCC	k	Branch if carry cleared	if(C==0) PC = PC + k + 1	None	1/2
BRSH	k	Branch if same or higher	if(C==0) PC = PC + k + 1	None	1/2
BRLO	k	Branch if lower	if(C==1) PC = PC + k + 1	None	1/2

Branch Instructions

Mnemonic	Operands	Description	Operation	Flags	Cycles
BRMI	k	Branch if minus	if(N==1) PC = PC + k + 1	None	1/2
BRPL	k	Branch if plus	if(N==0) PC = PC + k + 1	None	1/2
BRGE	k	Branch if greater than or equal (signed)	if(S==0) PC = PC + k + 1	None	1/2
BRLT	k	Branch if less than (signed)	if(S==1) PC = PC + k + 1	None	1/2
BRHS	k	Branch if half carry flag set	if(H==1) PC = PC + k + 1	None	1/2
BRHC	k	Branch if half carry flag cleared	if(H==0) PC = PC + k + 1	None	1/2
BRTS	k	Branch if T flag set	if(T==1) PC = PC + k + 1	None	1/2
BRTC	k	Branch if T flag cleared	if(T==0) PC = PC + k + 1	None	1/2
BRVS	k	Branch if overflow flag set	if(V==1) PC = PC + k + 1	None	1/2
BRVC	k	Branch if overflow flag cleared	if(V==0) PC = PC + k + 1	None	1/2
BRIE	k	Branch if interrupt enabled	if(I==1) PC = PC + k + 1	None	1/2
BRID	k	Branch if interrupt disabled	if(I==0) PC = PC + k + 1	None	1/2

Data Transfer Instructions

Mnemonic	Operands	Description	Operation	Flags	Cycles
MOV	Rd,Rr	Copy register	Rd = Rr	None	1
MOVW	Rd,Rr	Copy register pair	Rd+1:Rd = Rr+1:Rr, r,d even	None	1
LDI	Rd,K8	Load Immediate	Rd = K	None	1
LDS	Rd,k	Load Direct	Rd = (k)	None	2*
LD	Rd,X	Load Indirect	Rd = (X)	None	2*
LD	Rd,X+	Load Indirect and Post-Increment	Rd = (X), X=X+1	None	2*
LD	Rd,-X	Load Indirect and Pre-Decrement	X=X-1, Rd = (X)	None	2*
LD	Rd,Y	Load Indirect	Rd = (Y)	None	2*
LD	Rd,Y+	Load Indirect and Post-Increment	Rd = (Y), Y=Y+1	None	2*
LD	Rd,-Y	Load Indirect and Pre-Decrement	Y=Y-1, Rd = (Y)	None	2*
LDD	Rd,Y+q	Load Indirect with displacement	Rd = (Y+q)	None	2*
LD	Rd,Z	Load Indirect	Rd = (Z)	None	2*
LD	Rd,Z+	Load Indirect and Post-Increment	Rd = (Z), Z=Z+1	None	2*
LD	Rd,-Z	Load Indirect and Pre-Decrement	Z=Z-1, Rd = (Z)	None	2*
LDD	Rd,Z+q	Load Indirect with displacement	Rd = (Z+q)	None	2*
STS	k,Rr	Store Direct	(k) = Rr	None	2*

Data Transfer Instructions

Mnemonic	Operands	Description	Operation	Flags	Cycles
ST	X,Rr	Store Indirect	(X) = Rr	None	2*
ST	X+,Rr	Store Indirect and Post-Increment	(X) = Rr, X=X+1	None	2*
ST	-X,Rr	Store Indirect and Pre-Decrement	X=X-1, (X)=Rr	None	2*
ST	Y,Rr	Store Indirect	(Y) = Rr	None	2*
ST	Y+,Rr	Store Indirect and Post-Increment	(Y) = Rr, Y=Y+1	None	2
ST	-Y,Rr	Store Indirect and Pre-Decrement	Y=Y-1, (Y) = Rr	None	2
ST	Y+q,Rr	Store Indirect with displacement	(Y+q) = Rr	None	2
ST	Z,Rr	Store Indirect	(Z) = Rr	None	2
ST	Z+,Rr	Store Indirect and Post-Increment	(Z) = Rr, Z=Z+1	None	2
ST	-Z,Rr	Store Indirect and Pre-Decrement	Z=Z-1, (Z) = Rr	None	2
ST	Z+q,Rr	Store Indirect with displacement	(Z+q) = Rr	None	2
LPM	None	Load Program Memory	R0 = (Z)	None	3
LPM	Rd,Z	Load Program Memory	Rd = (Z)	None	3

Data Transfer Instructions

Mnemonic	Operands	Description	Operation	Flags	Cycles
LPM	Rd,Z+	Load Program Memory and Post-Increment	Rd = (Z), Z=Z+1	None	3
ELPM	None	Extended Load Program Memory	R0 = (RAMPZ:Z)	None	3
ELPM	Rd,Z	Extended Load Program Memory	Rd = (RAMPZ:Z)	None	3
ELPM	Rd,Z+	Extended Load Program Memory and Post Increment	Rd = (RAMPZ:Z), Z = Z+1	None	3
SPM	None	Store Program Memory	(Z) = R1:R0	None	-
ESPM	None	Extended Store Program Memory	(RAMPZ:Z) = R1:R0	None	-
IN	Rd,P	In Port	Rd = P	None	1
OUT	P,Rr	Out Port	P = Rr	None	1
PUSH	Rr	Push register on Stack	STACK = Rr	None	2
POP	Rd	Pop register from Stack	Rd = STACK	None	2

Bit and Bit-test Instructions

Mnemonic	Operands	Description	Operation	Flags	Cycles
LSL	Rd	Logical shift left	Rd(n+1)=Rd(n), Rd(0)=0, C=Rd(7)	Z,C,N,V,H,S	1
LSR	Rd	Logical shift right	Rd(n)=Rd(n+1), Rd(7)=0, C=Rd(0)	Z,C,N,V,S	1
ROL	Rd	Rotate left through carry	Rd(0)=C, Rd(n+1)=Rd(n), C=Rd(7)	Z,C,N,V,H,S	1
ROR	Rd	Rotate right through carry	Rd(7)=C, Rd(n)=Rd(n+1), C=Rd(0)	Z,C,N,V,S	1
ASR	Rd	Arithmetic shift right	Rd(n)=Rd(n+1), n=0,...,6	Z,C,N,V,S	1
SWAP	Rd	Swap nibbles	Rd(3..0) = Rd(7..4), Rd(7..4) = Rd(3..0)	None	1
BSET	s	Set flag	SREG(s) = 1	SREG(s)	1
BCLR	s	Clear flag	SREG(s) = 0	SREG(s)	1
SBI	P,b	Set bit in I/O register	I/O(P,b) = 1	None	2
CBI	P,b	Clear bit in I/O register	I/O(P,b) = 0	None	2
BST	Rr,b	Bit store from register to T	T = Rr(b)	T	1
BLD	Rd,b	Bit load from register to T	Rd(b) = T	None	1
SEC	None	Set carry flag	C =1	C	1
CLC	None	Clear carry flag	C = 0	C	1

Bit and Bit-test Instructions

Mnemonic	Operands	Description	Operation	Flags	Cycles
SEN	None	Set negative flag	N = 1	N	1
CLN	None	Clear negative flag	N = 0	N	1
SEZ	None	Set zero flag	Z = 1	Z	1
CLZ	None	Clear zero flag	Z = 0	Z	1
SEI	None	Set interrupt flag	I = 1	I	1
CLI	None	Clear interrupt flag	I = 0	I	1
SES	None	Set signed flag	S = 1	S	1
CLN	None	Clear signed flag	S = 0	S	1
SEV	None	Set overflow flag	V = 1	V	1
CLV	None	Clear overflow flag	V = 0	V	1
SET	None	Set T-flag	T = 1	T	1
CLT	None	Clear T-flag	T = 0	T	1
SEH	None	Set half carry flag	H = 1	H	1
CLH	None	Clear half carry flag	H = 0	H	1
NOP	None	No operation	None	None	1
SLEEP	None	Sleep	See instruction manual	None	1
WDR	None	Watchdog Reset	See instruction manual	None	1

Alcuni Compilatori e sistemi di sviluppo presenti sul mercato per i Microcontrollori AVR

- *AVRStudio4 della ATMEL*
- *AVR-GCC Free GNU*
- *CodeVision AVR*
- *ICC-AVR della ImageCraft*
- *FastAVR*
- *mikroBasic for AVR*
- ***BASCOM-AVR della MCSELEC***

Dove scaricare la documentazione e le Demo

www.delucagiovanni.com > Docs > Archimede

Il sito ufficiale Bascom:

www.mcselec.com

Il sito ufficiale Atmel:

www.atmel.com

Perché il BASCOM ?

- *Linguaggio ad alto livello, vicino al PLM-51 – è possibile scrivere anche in "asm" per creare librerie personalizzate compatte distribuibili e criptate*
- *Di facile intuizione – Sintassi dei comandi tra il "C standard" e il "BASIC"*
- *Linguaggio ad esecuzione sequenziale e strutturato con subroutine e funzioni*
- *Velocità nella produzione di firmware robusto ed affidabile (ottimo Time to Market)*
- *Grandissima varietà di micro AVR supportati dal compilatore 8/16/32 bit*
- *Presente sul mercato da oltre 15 anni*
- *Costo non elevato del pacchetto professionale*
- *Librerie disponibili per la gestione di periferiche: GLCD, ADC, DAC, IO ext, MEM, TCP-IP, USB*
- *Ottimizzazione del codice nativo grazie alla compilazione a passaggi multipli*
- *Possibilità di implementare un FileSystem compatibile DOS-FAT16/32*
- *Gestione diretta di HARDISK IDE, SD-CARD, e altri dispositivi simili*
- *Versioni del Software in continuo aggiornamento e implementazione nuovi comandi*
- *Possibile emulazione del firmware prodotto, con AVRStudio o PROTEUS-ISIS*
- *Moltissimi lavori professionali prodotti con questo pacchetto software*
- *Molta documentazione in rete e vari esempi esaustivi*
- *Forum attivo sul sito www.MSCELEC.com (registrazione richiesta)*
- *Supporto on-line professionale 24H24 (a pagamento)*

Perché AVR con Bascom

da una fonte autorevole

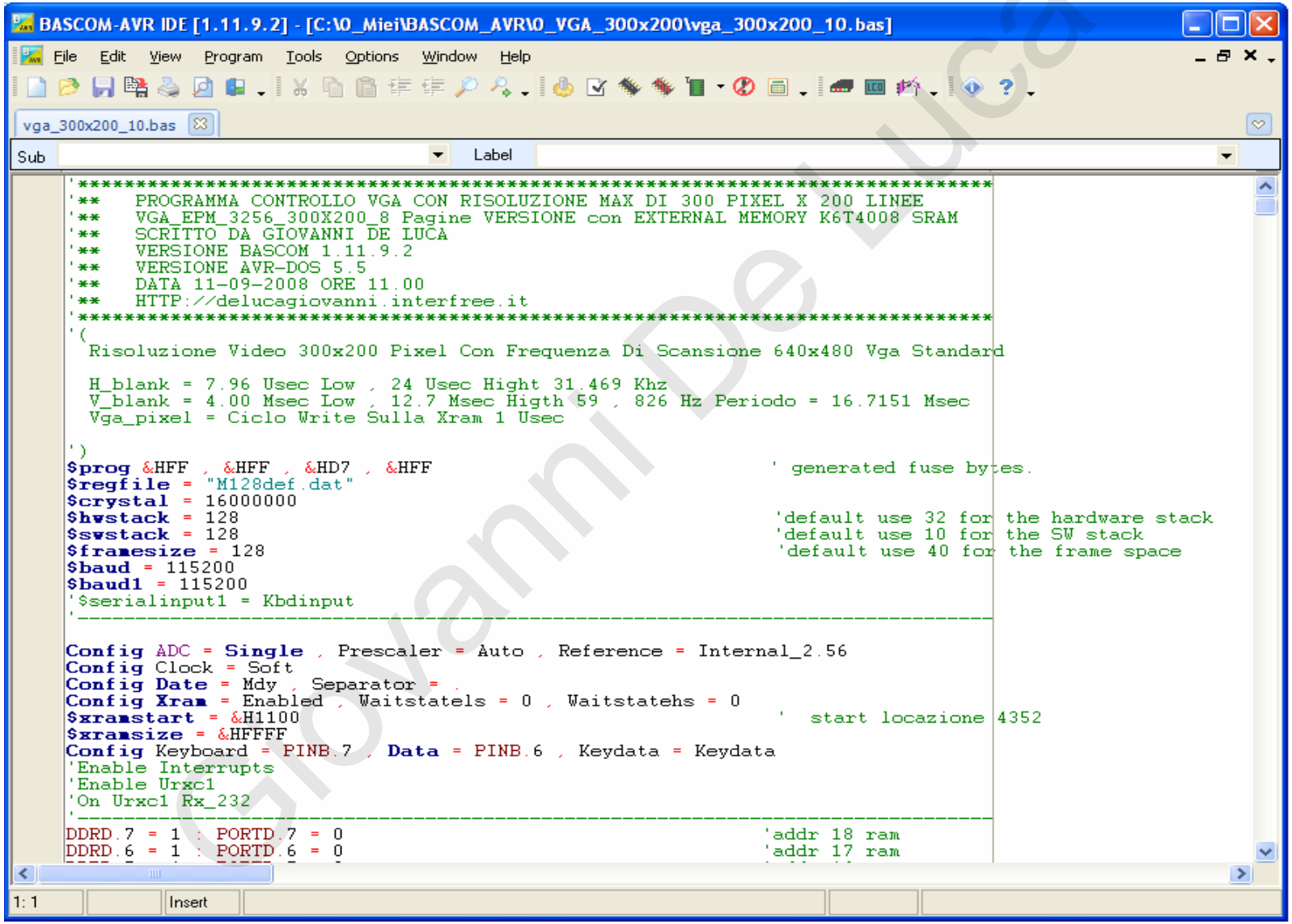
- *AVR is a family of 8-bit microcontrollers with a large range of variants differing in:*
  - size of program-memory (flash)
  - size of eeprom memory
  - number of I/O pins
  - number of on-chip features such as uart and adc
  - package forms

- *The smallest microcontroller is the ATTiny11 with 1k flash and 6 I/O pins. The largest is the ATxMEGA256x with 256k flash, 54 I/O pins and lots of on-chip features. All AVR controllers have the same RISC-like instruction set, enabling fairly easy porting of Bascom programs between microcontroller types.*
- ***(Tutti i controllori AVR hanno lo stesso set di istruzioni RISC-like, che consente il porting abbastanza facile dei programmi di Bascom tra i tipi di microcontroller)***

- *They execute one instruction per clock-cycle making them appreciably faster than comparable 8-bit 4 clock-cycles-per-instruction Microchip PIC controllers .*

  (Essi eseguono una sola istruzione per ciclo di clock li rende sensibilmente più veloce rispetto ai tradizionali 8-bit 4 cicli di clock-per-istruzione Microchip PIC controller)

L’ambiente di sviluppo IDE



Arduino “Blink”

```
int ledPin = 13;
void setup()
    {
      pinMode(ledPin, OUTPUT);
    }

void loop()
    {
      digitalWrite(ledPin, HIGH);
      delay(1000);
      digitalWrite(ledPin, LOW);
      delay(1000);
    }
```

'1112 bytes

```
$Include "DuinoLib.inc"
Config Pin13 = Output

Do
  Set Out13
  WaitmS 1000
  Reset Out13
  WaitmS 1000
Loop
```

'230 bytes

Arduino Example: 'Smoothing'

```
#define NUMREADINGS 10
int readings[NUMREADINGS];
int index = 0;
int total = 0;
int average = 0;
int inputPin = 0;
void setup()
      {
        Serial.begin(9600);
        for (int i = 0; i < NUMREADINGS;i++)
         readings[i] = 0;
      }

void loop()
      {
        total - = readings[index];
        readings[index] =
      analogRead(inputPin);
        total + = readings[index];
        index =(index + 1);
        if(index >= NUMREADINGS)
          index = 0;
           average = total / NUMREADINGS;
        Serial.println(average);
      }
```

'2,838 bytes

```
$Include "DuinoLib.inc"
$External AnaRead
Dim Index as Byte , Value As Word
Dim Total As Word , Average as Word
Dim Readings(10) As Word

Const NumReadings = 10
Index = 1

Do
   Total = Total - Readings(Index)
   AnaRead 5 , Value
   Readings(Index) = Value
   Total = Total + Value
   Incr Index
   If Index > NumReadings Then Index = 1
   Average = Total / NumReadings
   Print Average
Loop
```

'870 bytes

Quante istruzioni dobbiamo conoscere per scrivere un Programma FW ?

- *133 - Istruzioni Assembly AVR*
- *405 - Istruzioni, nel Bascom AVR nella versione 2.0.5.0*

Primi passi con il BAS-AVR

Esempio di configurazione base del dispositivo ATMEGA128

```
$prog &HFF , &HFF , &HD7 , &HFF
$regfile = "M128def.dat"
$crystal = 16000000
$hwstack = 128
$swstack = 128
$framesize = 128
$baud = 115200
$baud1 = 115200
```

La potenza del compilatore BAS-AVR

```
DIM X as byte
    Print X
End
```

Linguaggio Assembly

```
        .DSEG
.ORG    256
x:                      .Byte    1

        .CSEG
.ORG    0

_Reset:
        ldi     yl,Low(RAMEND)
        out     SPL,yl
        ldi     yh,high(RAMEND)
        out     SPL+1,yh
        sbiw    yl,32
        ldi     zl,0x18
        out     UCSR0B,zl
        ldi     zh,high(8)
        ldi     zl,Low(8)
        out     UBRR0L,zl
        sts     UBRR0H,zh


;****** USERS BASIC CODE **********************

;-Line--0013----Print x--
        lds     zl,x
        call    _B2Str
        call    _PrBW
        call    _PCL

;-Line--0014----End--
L0000:
        jmp L0000

;****** End OF USER BASIC CODE *****************
```

```
;////// Print Byte & Word /////////////////
_PrBW:  ld      r24,Z+
        tst     r24
        breq    _PBW1
        rcall   _Pch
        rjmp    _PrBW
_PBW1:  ret

;////// Print Cr, Lf & any char//////////////
_PCL:   ldi     r24,0x0d
        rcall   _Pch
        ldi     r24,0x0a
_Pch:   sbis    UCSR0A,5
        rjmp    _Pch
        out     UDR0,r24
        ret

;////// ByteToStr /////////////////////////
_B2str: clr     zh
        clt
        push    yl
        push    yh
        st      -Y,zh
_N2str: ldi     r21,0x08
        Sub     r22,r22
_N2stl: lsr     zh
        rol     zl
        rol     r22
        rol     zh
        cpi     r22,0x0a
        brcs    _N2st2
        sbci    r22,0x0a
        inc     zl
_N2st2: dec     r21
        brne    _N2stl
        subi    r22,-0x30
        st      -Y,r22
        tst     zl
        brne    _N2str
_N2st3: mov     zl,yl
        mov     zh,yh
        pop     yh
        pop     yl
        ret

;System Global Variables: 0  bytes
;User   Global Variables: 1  bytes
```