Watchdog I2C – Watchdog con interfaccia I2C

Un watchdog dal funzionamento tradizionale oppure avanzato, con interfaccia I2C/SMBus

Si tratta di un watchdog per microprocessori e PC basato su un microcontrollore PIC della Microchip.
La sua tipica applicazione consiste nel supervisionare il corretto funzionamento di un sistema a microprocessore, generando un impulso di reset qualora venga rilevato il blocco del sistema.
In genere un watchdog dispone di un piedino di input che il sistema deve periodicamente far cambiare di stato, e di un piedino di output dal quale viene ricavato l'impulso di reset per il sistema: il nostro watchdog dispone dei seguenti ingressi ed uscite:

Pin 1	Vcc	Alimentazione 5V
Pin 2	OUTRES	Reset (attivo basso)
Pin 3	OUTNMI/WDCLR	NMI (attivo alto) oppure Azzera Watchdog
Pin 4	ALARM	Allarme/Power down (attivo basso)
Pin 5	GPO	Out generico (attivo alto)
Pin 6	I2CCLK	Clock I2C
Pin 7	I2CDATA	Dati I2C
Pin 8	GND	Massa – 0V

Al piedino 3 va applicato il segnale proveniente dal sistema, che indica che il programma sta girando correttamente. Se per un periodo di tempo superiore a quello impostato (dalla fabbrica o mediante bus I2C) il piedino 3 rimane bloccato, il watchdog genera un'impulso negativo della durata di 200mS sull'uscita OUTRES, un impulso della durata di 5 secondi sul piedino ALARM, e porta alto il piedino GPO; quest'ultimo resterà nello stato attivo fino allo spegnimento della macchina (o alla ricezione di un apposito comando dall'interfaccia I2C). Successivamente il ciclo si ripete da capo.

Ogni aspetto del watchdog fin qui descritto può essere personalizzato in fase di fabbricazione oppure mediante l'invio di comandi sul bus I2C.
Il bus I2C è un'interfaccia standard per dispositivi elettronici. Sviluppato inizialmente dalla Philips ed utilizzato nei televisori, si è in seguito diffuso come standard per il dialogo tra microprocessori e periferiche in ogni tipo di computer.
Sostanzialmente si tratta di un bus gestito da uno o più master (ma uno solo alla volta può avere il controllo del bus), sul quale si possono situare fino a 127 dispositivi. Fisicamente utilizza due fili (clock e dati) a livelli TTL/CMOS, sui quali corrono dei segnali dalla temporizzazione ben definita: il periodo di clock va da circa 8 a circa 60uSec (si veda la documentazione Philips per approfondimenti).
Il nostro watchdog risponde all'indirizzo 46H, che diventa 47H per le richieste di lettura (infatti il bit meno significativo indica se siamo in presenza di una scrittura master->slave (bit a 0) o di una lettura (bit a 1))

Il watchdog dispone di 3 registri interni, leggibili e scrivibili:
R0:

b1 b0:	WD attivo (00=disattivato,01 scatta su inattività, 10 scatta su attività troppo rapida*, 11 entrambe)
b3 b2:	tipo output (00=reset-attesa(impulso)-ricomincia, 01=reset-attesa(impulso)-stop, 11=reset(resta alto)-stop. 10=NON VALIDO!)
b4:	1=abilita un pin al resetWD via hardware (metodo classico)
b5:	1=ALARM diventa pre-alarm (attivo basso, fisso) a metà del time-out*
b6:	1=tempi lunghi (1 sec...255sec), 0=tempi brevi 10mS...2550mSec
b7:	1=il WD è scattato (sola lettura, la scrittura di CFG lo resetta!)

R1:

b0:	1= -OUTRES abilitato
b1:	(riservato*)
b2:	1= OUTNMI abilitato (100mSec) (se CFG0<4> = 0)
b3:	1= -ALARM abilitato (segue CFG0<b2..3>, con impulso=5sec)
b4:	1= GPO pilotato dal WD (va alto, fisso); 0: GPO pilotato da b5-b6
b5:	(se b4=0) 0= GPO è pilotato da b6; 1=abilita il pin GPO al lampeggio (0.5Hz) di un led

b6:	(se b4&b5=0) stato del led (pin GPO)

R2:

durata, espressa in decine di millisecondi oppure in secondi, a seconda dello stato di R0<6>

* = *future implementazioni.*

Il watchdog risponderà positivamente alle operazioni di scrittura e lettura standard, ma il suo funzionamento è ottimizzato per rispettare le estensioni SMBus (system management bus), che vediamo nel dettaglio.

Scrittura:

Start	Indirizzo	SubIndir	Byte 0	Byte 1	Byte 2	Byte 3	Stop		
	46H	0	N/A	N/A	N/A	N/A			Nessun effetto
	46H	1	R0	R1	R2				Imposta R0..2
	46H	2	R2						Imposta R2 (durata)
	46H	3	(dummy)						Reset Watchdog

Lettura:

Start	Indirizzo	SubIndir	Byte	Stop		
	47H	0	8H, 8 byte ASCII ID			Legge l'ID
	47H	1	3H, R0, R1, R2			Legge R0..2
	47H	2	3H, contatori 64uS, 10mS, 1S			Legge i 3 contatori interni
	47H	3	3H, livello dei pin			Legge lo stato dei pin

In fase di scrittura si definisce che, dopo l'indirizzo e l'ACK da parte del dispositivo, viene selezionato un Comando (o sub-indirizzo) inviando un byte compreso tra 0 e 3. Successivamente vengono inviati i byte relativi, che il dispositivo confermerà uno ad uno (fino a un massimo di 4) e scriverà nei registri indicati.
In fase di lettura si utilizza un approccio simile: viene inviato un Comando (dopo che l'indirizzo è stato riconosciuto) compreso tra 0 e 3 ed a questo punto il dispositivo risponde con il numero di byte che ha intenzione di spedire, facendo poi seguire i dati effettivi. Questo tipo di lettura viene in realtà effettuato mediante l'estensione denominata "BLOCK_READ" al protocollo I2C, che in pratica definisce quanto segue:
l'operazione di lettura viene tradotta in una prima operazione di scrittura, con la quale si invia l'indirizzo e il sub-indirizzo;
quindi, senza interrompere la comunicazione (ossia evitando lo STOP ma inviando un nuovo segnale di START, denominato RESTART) il master inizia un'operazione di lettura per tutti i byte che il dispositivo ha intenzione di inviargli.

A causa di questa implementazione, il watchdog sarà compatibile in scrittura con le operazioni di Write semplice del protocollo I2C, ma in lettura l'operazione di Read semplice restituirà sempre il solo byte "8H" seguito dalla stringa di identificazione.

Per ulteriori approfondimenti sulle estensioni al protocollo I2C, come previste dal protocollo Intel PIIX4 (BLOCK_READ ed altro), si veda l'Appendice I (interfacciamento al SMBus del chip VIA VT82C686 – South Bridge).
Per altri esempi di utilizzo dell'I2C si veda inolre l'Appendice II (altri esempi di periferiche I2C).

Torniamo all'analisi del funzionamento del watchdog, prendendo in considerazione i 3 registri di configurazione.

I bit 0 e 1 del registro R0 consentono di disattivare il watchdog, o di attivarlo in maniera che rilevi la prolungata inattività del sistema: come abbiamo visto, il sistema può azzerare il contatore del watchdog sia cambiando di stato il pin WDCTRL (purché esso sia attivo – v. seguito) sia inviando il comando 3 sul bus I2C. E' prevista in futuro l'aggiunta di una funzione di watchdog in caso di "iperattività", ossia qualora il sistema stia manifestando un'attività troppo rapida (segno probabile di un funzionamento al di fuori dei parametri normali).

I bit 2 e 3 del registro R0 consentono di specificare diversi tipi di funzionamento del watchdog. La modalità di default è “contatempo-reset-ricomincia”, il che significa che il watchdog terrà sotto controllo il sistema rilevandone gli eventuali malfunzionamenti, al manifestarsi dei quali farà scattare la fase di Reset (sui pin configurati), per poi ricominciare da capo. E’ possibile fare in modo che il ciclo non si ripeta, ossia che dopo esser entrato in una fase di reset il watchdog non ne possa causare altre se prima non viene ripristinato tramite I2C. Come ulteriore variazione è possibile far sì che oltre a non consentire nuovi Reset, il watchdog mantenga indefinitamente alto il pin ALARM.
Naturalmente la modalità 00 è la più sicura in quanto continuerà a far ripartire il sistema fino a quando questo non ricomincia a funzionare normalmente (il primo reset potrebbe non aver avuto effetto per svariati motivi): ci sono però situazioni in cui le modalità 1x sono preferibili.
Il bit 4 di R0, quando è a 1, consente l’azzeramento del watchdog tramite il pin 3. Se questo bit è a 0, tale funzione non è disponibile (l’azzeramento può avvenire solo via I2C). In genere questo bit viene configurato in fabbrica per avere dei watchdog dal funzionamento ben definito.
Il bit 5 prevede l’utilizzo dell’uscita ALARM come indicatore di “pre-allarme”, che scatta quando il watchdog ha raggiunto la metà del suo tempo totale prima di procedere al reset. Può essere utile per avvisare un operatore del malfunzionamento del sistema prima che il watchdog lo faccia ripartire. (funzione futura)
Il bit 6 di R0 è molto importante in quanto permette di scegliere due gamme di tempi. Se vale 0, potremo disporre di tempi di time-out compresi tra 10 e 2550 millisecondi; se vale 1, andremo da 1 secondo a 255 secondi (4’15”)
Il bit 7 di R0 è un indicatore a sola lettura del fatto che il watchdog è scattato in almeno un’occasione dall’ultima volta che è stato inizializzato. Questo indicatore serve soprattutto nel caso in cui si utilizzi una delle modalità 1x del funzionamento del watchdog (ossia ciclo che non si ripete – v. sopra)

Il registro R1 consente di attivare oppure disattivare i pin di uscita che desideriamo usare.
I bit 0 e 1 agiscono sull’uscita OUTRES (pin 2). Il bit 0 indica se questa uscita potrà andare a livello basso oppure no; il bit 1 è per ora riservato a future implementazioni.
Il bit 2 indica se sull’uscita OUTNMI (pin3) si verificherà un impulso di 100mSec al raggiungimento della fase di reset. Se questo bit è a 0, il pin OUTNMI rimane sempre a livello 0.
Il bit 3 indica se l’uscita ALARM (pin4) andrà a livello basso per 5 secondi (o indefinitamente, a seconda della configurazione di R0<2..3>) Se questo bit è a 0, il pin ALARM rimane sempre a livello 1.
I bit 4..6 specificano il comportamento dell’uscita GPO (pin 5). Se b4 vale 1, il pin GPO rimane normalmente a livello basso, salvo portarsi a 1 quando il watchdog entra nella fase di reset; questa uscita NON segue la configurazione di R0<2..3>, ma resta alta fino al reset via I2C.
Se b4 vale 0, l’uscita GPO viene pilotata dai bit 5 e 6. Se b5 vale 0, l’uscita GPO vale il valore scritto in b6; se b5 vale 1, l’uscita GPO oscilla con frequenza 0.5Hz (adatta per il lampeggìo di un led di buon funzionamento).

Il registro R2 indica il tempo di time-out del watchdog, espresso in decine di millisecondi (se R0<6> vale 0) oppure in secondi (se R0<6> vale 1).

Va notato che, durante la fase di Reset, il watchdog non accetterà nessun comando di scrittura da parte del master I2C. Questo per evitare situazioni particolari in cui il sistema, evidentemente diventato instabile al punto da necessitare di un reset, invia comandi a caso al watchdog eventualmente causandone errate temporizzazioni sui piedini di uscita o addirittura il mancato funzionamento previsto. Questo periodo di tempo dura da 1 a 6 secondi dopo che il periodo di time-out è scaduto.

Si tenga inoltre presente che anche l’invio dei comandi 1 e 2 sul bus I2C provoca l’azzeramento dei contatori interni del watch-dog.

Infine va notato come questo watchdog possa essere alimentato sia a 3.3V che a 5V. Anche a 5Vcc di alimentazione, il dispositivo rimane compatibile a livello di segnali con chip alimentati a 3.3V.

APPENDICE I

VIA South Bridge VT82C686

Come indicato dai data-sheet, il controller dell'I2C-SMBus del chip VIA 82C686 viene situato nello spazio di I/O a 16 bit specificando un opportuno valore nei registri 93..90 della funzione 4 dei registri di configurazione del bus PCI. Tale valore viene in generale impostato dal BIOS e riconosciuto poi da Windows ("risorse della scheda di sistema"): sia dentro Windows che in modalità MS-DOS l'indirizzo utilizzato dalle macchine MVP4 è 5000H.

System Management Bus I/O-Space Registers
The base address for these registers is defined in Rx93-90 of the Function 4 PCI configuration registers. The System Management Bus I/O space is enabled for access by the system if RxD2[0] = 1.

(pag. 95)

System Management Bus-Specific Configuration Registers	
Offset 93-90 – SMBus I/O Base	**RW**
31-16 Reserved	always reads 0
15-4 I/O Base (16-byte I/O space)	default = 00h
3-0 Fixed	always reads 0001b
Offset D2 – SMBus Host Configuration	**RW**
7-4 Reserved	always reads 0
3 SMBus Interrupt Select	
0 SMI	default
1 SCI	
2 Reserved	always reads 0
1 SMBus I RQ	
0 Disable	default
1 Enable	
0 SMBus Host Controller Enable	
0 Disable SMB controller functions	default
1 Enable SMB controller functions	
Offset D3 – SMBus Host Slave Command	**RW**
7-0 SMBus Host Slave Command Code	default=0
Offset D4 – SMBus Slave Address for Port 1	RW
7-0 SMBus Slave Address for Port 1	default=0
Bit-0 must be set to 0 for proper operation	
Offset D5 – SMBus Slave Address for Port 2	**RW**
7-0 SMBus Slave Address for Port 2	default=0
Bit-0 must be set to 0 for proper operation	
Offset D6 – SMBus Revision ID	**RO**
7-0 SMBus Revision Code	

All'interno di questo spazio, troviamo un certo numero di registri, alcuni dei quali non ci interessano in quanto trattano la trasmissione slave-host. Prenderemo in considerazione quelli che riguardano la comunicazione iniziata dall'Host che è poi il chip VIA stesso.
Del registro 0 ci interessa soltanto il bit meno significativo, che indica se il controller è attualmente impegnato oppure no. Gli altri bit hanno a che fare con la generazione di un interrupt da parte dell'SMBus controller, cosa che attualmente non è usata.
Nel registro 2 troviamo il bit 6 che è molto importante in quanto causa l'esecuzione del comando impostato (va quindi attivato dopo aver preparato gli altri registri) ed il tipo di transazione che desideriamo effettuare. Come spiegato nel corso di questa descrizione, la modalità che ci interessa particolarmente è la "BLOCK_READ_OR_WRITE" (101b). Anche le altre sono utilizzabili, ma solo in maniera limitata ed in pratica inutilizzabile.
Nei registri 3 e 4 troviamo rispettivamente il comando e l'indirizzo al quale inviarlo.
Il registri 5 e 6 sono due registri in cui vengono posti il primo oppure i primi due byte inviati o ricevuti sul bus dopo il comando.
Il registro 7 viene usato per accedere ad un buffer da 32 byte (massima lunghezza di una transazione) interno al controller, il cui puntatore viene incrementato automaticamente ad ogni lettura o scrittura, ed azzerato ad ogni lettura del registro 2.

(pag. 105)

I/O Offset 00 – SMBus Host Status	**RWC**
7-5 Reserved	always reads 0
4 Failed Bus Transaction	RWC

0 SMBus interrupt not caused by failed bus transaction default
1 SMBus interrupt caused by failed bus transaction. This bit may be set when the KILL bit (I/O Rx02[1]) is set and can be cleared by writing a 1 to this bit position.

3 Bus Collision RWC
0 SMBus interrupt not caused by transaction collision default
1 SMBus interrupt caused by transaction collision. This bit is only set by hardware and can be cleared by writing a 1 to this bit position.

2 Device Error RWC
0 SMBus interrupt not caused by generation of an SMBus transaction error default
1 SMBus interrupt caused by generation of an SMBus transaction error (illegal command field, unclaimed host-initiated cycle, or host device timeout). This bit is only set by hardware and can be cleared by writing a 1 to this bit position.

1 SMBus Interrupt RWC
0 SMBus interrupt not caused by host command completion default
1 SMBus interrupt caused by host command completion. This bit is only set by hardware and can be cleared by writing a 1 to this bit position.

0 Host Busy RO
0 SMBus controller host interface is not processing a command default
1 SMBus host controller is busy processing a command. None of the other SMBus registers should be accessed if this bit is set.

I/O Offset 02h – SMBus Host Control **RW**

7 Reserved always reads 0

6 Start always reads 0
0 Writing 0 has no effect default
1 Start Execution of Command
Writing a 1 to this bit causes the SMBus controller host interface to initiate execution of the command programmed in the SMBus Command Protocol field (bits 4-2). All necessary registers should be programmed prior to writing a 1 to this bit. The Host Busy bit (SMBus Host Status Register bit-0) can be used to identify when the SMBus controller has completed command execution.

5 Reserved always reads 0

4-2 SMBus Command Protocol
000 Quick Read or Write default
001 Byte Read or Write
010 Byte Data Read or Write
011 Word Data Read or Write
100 Reserved
101 Block Read or Write
110 Reserved
111 Reserved

1 Kill Transaction in Progress
0 Normal host controller operation default
1 Stop host transaction currently in progress.
Setting this bit also sets the FAILED status bit (Host Status bit-4) and asserts the interrupt selected by the SMB Interrupt Select bit (Function 4 SMBus Host Configuration Register RxD2[3]).

0 Interrupt Enable
0 Disable interrupt generation default
1 Enable generation of interrupts on completion of the current host transaction.

I/O Offset 03h – SMBus Host Command **RW**

7-0 SMBUS Host Command default = 0
This field contains the data transmitted in the command field of the SMBus host transaction.

I/O Offset 04h – SMBus Host Address **RW**

The contents of this register are transmitted in the address field of the SMBus host transaction.

7-1 SMBUS Address default = 0
This field contains the 7-bit address of the targeted slave device.

0 SMBUS Read or Write
0 Execute a WRITE command default
1 Execute a READ command

I/O Offset 05h – SMBus Host Data 0 **RW**

The contents of this register are transmitted in the Data 0 field of SMBus host transaction writes. On reads, Data 0 bytes are stored here.

7-0 SMBUS Data 0 default = 0
For Block Write commands, this field is programmed with the block transfer count (a value between 1 and 32). Counts of 0 or greater than 32 are undefined. For Block Read commands, the count received from the SMBus device is stored here.

I/O Offset 06h – SMBus Host Data 1 **RW**

The contents of this register are transmitted in the Data 1 field of SMBus host transaction writes. On reads, Data 1 bytes are stored here.

7-0 SMBUS Data 1 default = 0

I/O Offset 07h – SMBus Block Data **RW**

Reads and writes to this register are used to access the 32-byte block data storage array. An internal index pointer is used to address the array. It is reset to 0 by reads of the SMBus Host Control register (I/O Offset 2) and incremented automatically by each access to this register. The transfer of block data into (read) or out of (write) this storage array during an SMBus transaction always starts at index address 0.

7-0 SMBUS Block Data default = 0

In pratica, per effettuare una scrittura nel watchdog, occorre effettuare le seguenti operazioni:

```
ba=5000H     'base registri
```

```
WHILE (INP(ba) AND 1) <> 0: WEND   ‘attendere che l’host sia
disponibile
OUT ba + 3, 1     ‘ preparare il comando
OUT ba + 4, 46H   ‘ indirizzo in scrittura
OUT ba + 5, 3     ‘ numero di byte che scriviamo
CALL setblock(65H, 32H, 240)    ‘ scrivere nel buffer da 32 byte i 3
valori che ci servono
tipoOp = 101B   ‘ tipo operazione = BLOCK_WRITE_OR_READ
OUT ba + 2, tipoOp * 4      ‘ mi preparo...
OUT ba + 2, (tipoOp * 4) OR 40H    ‘ ...poi eseguo!
```

Questa operazione scrive i tre valori 65H,32H e 240 (watchdog attivo con ciclo continuo, time-out=4 minuti, led lampeggiante) nei registri di configurazione del watchdog (comando 1)

Mentre, per una lettura:

```
ba=5000H    ‘base registri
WHILE (INP(ba) AND 1) <> 0: WEND   ‘attendere che l’host sia
disponibile
OUT ba + 3, 0     ‘ preparare il comando
OUT ba + 4, 47H   ‘ indirizzo in lettura
tipoOp = 101B   ‘ tipo operazione = BLOCK_WRITE_OR_READ
OUT ba + 2, tipoOp * 4      ‘ mi preparo...
OUT ba + 2, (tipoOp * 4) OR 40H    ‘ ...poi eseguo!
’ in ba + 5 troviamo il numero di byte letti
’ e questi byte sono stati messi nel buffer, accessibile con R7
```

Questa operazione legge l’ID del dispositivo (comando 0)

La funzione setblock è definita molto semplicemente in questi termini:

```
SUB setblock (d0, d1, d2)    ‘ aggiungere altri parametri se serve!
SHARED ba
i = INP(ba + 2)  ‘ reset block_data buffer
OUT ba + 7, d0
OUT ba + 7, d1
OUT ba + 7, d2
END SUB
```

Mentre un’analoga funzione ci permetterà di estrarre i dati ricevuti dal buffer:

```
SUB dumpblock
SHARED ba
i = INP(ba + 2)    ‘ reset block_data buffer
PRINT "buffer: "; : FOR i = 0 TO 7: PRINT HEX$(INP(ba + 7)); "  "; :
NEXT: PRINT
END SUB
```

APPENDICE II

Nei computer MVP4 il bus I2C gestisce altre due periferiche: le memorie DIMM e il generatore di clock.

Le memorie DIMM incorporano infatti una piccola EEPROM seriale che provvede a comunicare ai PC di nuova generazione (all'incirca dal PII in poi) i parametri chiave delle memorie dinamiche che le compongono: numero di celle di memoria disponibili, temporizzazioni, struttura e/o geometria.
La memoria utilizzata è in genere una 24LCxx della Microchip (anche STM) che implementa il protocollo I2C.
Vediamo un piccolo estratto dal datasheet:

> The Microchip Technology Inc. 24AA00/24LC00/24C00 (24xx00*) is a 128-bit Electrically Erasable PROM memory organized as 16 x 8 with a 2-wire serial interface.

FIGURE 5-1: CONTROL BYTE FORMAT

Start Bit	Device Select Bits				Don't Care Bits			Read/Write Bit	Acknowledge Bit
S	**1**	**0**	**1**	**0**	**X**	**X**	**X**	**R/W**	**ACK**
	Slave Address								

Come si può notare, viene usato un particolare accorgimento "logico" per poter sistemare sul bus fino a 8 memorie: il byte (o meglio, i 7 bit costituenti l'indirizzo) viene spezzato in una parte fissa detta "device identifier" (infatti nessun altro dispositivo sul bus potrà avere un indirizzo che inizia con 0AH) e una parte variabile. Questa parte variabile verrà decodificata da una logica a bordo della scheda madre, la quale assegnerà per esempio l'indiizzo 0 dell'array di memorie al primo slot per DIMM, l'indirizzo 1 al secondo ecc. In questa maniera ogni slot delle memorie DIMM potrà essere indirizzato da un diverso indirizzo all'interno del pool di indirizzi assegnato a queste memorie EEPROM.

Per il resto, come possiamo vedere in un altro stralcio del data-sheet, le operazioni di lettura e scrittura da/verso la EEPROM utilizzano comandi I2C già noti:

FIGURE 7-2: BYTE WRITE

Bus activity master	Start	Control Byte									Word Address									Data									Stop
SDA line Bus activity	**S**	**1**	**0**	**1**	**0**	**X**	**X**	**X**	**0**	**ACK**	**X**	**X**	**X**	**X**					**ACK**									**ACK**	**P**

FIGURE 8-1: CURRENT ADDRESS READ

Bus activity master	Start	Control Byte									Data									Stop
SDA line Bus activity	**S**	**1**	**0**	**1**	**0**	**X**	**X**	**X**	**1**	**ACK**									**NO ACK**	**P**

FIGURE 8-2: RANDOM READ

Bus activity master	Start	Control Byte									Word Address									Start	Control Byte									Data (n)									Stop
SDA line Bus activity	S	1	0	1	0	X	X	X	0	ACK	X	X	X	X					ACK	S	1	0	1	0	X	X	X	1	ACK									NCACK	P

FIGURE 8-3: SEQUENTIAL READ

Bus activity master	Start	Control Byte									Data n									Data n+1									...	Data n+x									Stop
SDA line Bus activity	S	1	0	1	0	X	X	X	0	ACK									ACK									ACK									NACK	P

Il generatore di clock utilizzato è un **ICS9248-101** della Integrated Circuit Systems. Questo componente è in grado di generare tutti i segnali di clock richiesti per il funzionamento di un sistema a microprocessore di classe Pentium II con bus di sistema a 66, 100 e 133MHz. Gestisce inoltre il sistema dello “Spread-Spectrum” per la riduzione delle spurie EMI.
Senza soffermarci più di tanto sulle caratteristiche proprie di questo dispositivo, per le quali si rimanda al data-sheet, osserviamo la sua interfaccia I2C:

The **ICS9248-101** is the single chip clock solution for Notebook designs using the VIA Apollo Pro 133 style chipset.
It provides all necessary clock signals for such a system.

General I 2 C serial interface information
The information in this section assumes familiarity with I 2 C programming. For more information, contact ICS for an I 2 C programming application note.

How to Write:	How to Read:
• Controller (host) sends a start bit.	• Controller (host) will send start bit.
• Controller (host) sends the write address D2 (H)	• Controler (host) sends the read address D3 (H)
• ICS clock will ***acknowledge***	• ICS clock will ***acknowledge***
• Controller (host) sends a dummy command code	• ICS clock will send the ***byte count***
• ICS clock will ***acknowledge***	• Controller (host) acknowledges
• Controller (host) sends a dummy byte count	• ICS clock sends first byte ***(Byte 0) through byte 5***
• ICS clock will ***acknowledge***	• Controller (host) will need to acknowledge each byte
• Controller (host) starts sending first byte (Byte 0) through byte 5	• Controller (host) will send a stop bit
• ICS clock will ***acknowledge*** each byte ***one at a tim***e.	•
• Controller (host) sends a Stop bit	•

How to Write:		How to Read:	
Controller (Host)	**ICS (Slave/Receiver)**	**Controller (Host)**	**ICS (Slave/Receiver)**
Start Bit		Start Bit	
Address D2(H)		Address D3(H)	
	ACK		**ACK**
Dummy Command Code			**Byte Count**
	ACK	ACK	
Dummy Byte Count			**Byte 0**
	ACK	ACK	
Byte 0			**Byte 1**
	ACK	ACK	
Byte 1			**Byte 2**
	ACK	ACK	
Byte 2			**Byte 3**
	ACK	ACK	
Byte 3			**Byte 4**
	ACK	ACK	
Byte 4			**Byte 5**
	ACK	ACK	
Byte 5		Stop Bit	
	ACK		
Stop Bit			

Notes:
1. The ICS clock generator is a slave/receiver, I 2 C component. It can read back the data stored in the latches for verification. **Read-Back will support Intel PIIX4 "Block-Read" protocol**.
2. The data transfer rate supported by this clock generator is 100K bits/sec or less (standard mode)
3. The input is operating at 3.3V logic levels.

4. The data byte format is 8 bit bytes.
5. To simplify the clock generator I 2 C interface, the protocol is set to use only **"Block-Write**s" from the controller. The bytes must be accessed in sequential order from lowest to highest byte with the ability to stop after any complete byte has been transferred. The Command code and Byte count shown above must be sent, but the data is ignored for those two bytes. The data is loaded until a Stop sequence is issued.
6. At power-on, all registers are set to a default condition, as shown.

Esiste dunque un solo comando, sia in ricezione che in trasmissione; in entrambi i casi i paramteri vengono letti e scritti in blocchi da 6 byte che configurano le frequenze e le attivazioni dei diversi segnali di clock generati dal chip.
Non resta da aggiungere molto altro: il dispositivo si comporta esattamente secondo le specifiche BLOCK_READ e BLOCK_WRITE, analogamente al nostro Watchdog.

ADPM Synthesis
elettronica per l'automazione

ADPM Synthesis s.a.s. di Greggio D. & Vino P.
Sede legale, uffici e laboratorio:
via Rivalta 39 - 10141 Torino
Telefono: +39/011/336768-3823290-9137684
Fax: +39/011/336768
http://www.geocities.com/adpm99/hardsoft