ZR16S08 Datasheet

Rua Q, s/n - Prédio 67 - Campus da UFSM
CEP 97.105-900 - Santa Maria, RS - Brasil
contato@smdh.org

1 Visão geral

O ZR16S08 é um microcontrolador de baixo custo que enfatiza a integração analógico/digital, proporcionando uma ótima relação custo/desempenho.

Este microcontrolador possui um microprocessador de 8 bits, memória de programa de 1024 palavras, oscilador e regulador de tensão integrados, bem como interfaces de entrada/saída digitais e analógicas.

Ele conta também com periféricos analógicos e digitais, como Conversor Analógico para Digital, Sensor Capacitivo, Comparador Analógico, *Timer* e *Watchdog*.

2 Características

Microprocessador

- 24 instruções.
- 16 registradores, pilha de 4 posições de contexto para armazenamento.
- 8 bits de largura do barramento de dados.

Memória

- Memória de programa EEPROM de 1024 x 16 bits.
- Memória de dados SRAM de 256 x 8 bits.

Periféricos digitais

- *Timer* de 16 bits, base de tempo de $1\mu s$.
- *Watchdog* com tempo de expiração configurável.

Periféricos analógicos

- Conversor analógico para digital com 10 bits de resolução (ADC).
- Sensor Capacitivo, Comparador de 2 entradas analógicas ou de uma entrada analógica contra uma referência.
- Conversor digital para analógico (DAC), consiste em 3 saídas de corrente com capacidade de 30V/20mA.

Clock e alimentação

- Oscilador interno de 4MHz.
- Regulador de tensão com entrada entre 5 e 28V.

Interfaces externas

- Possui 16 pinos, dos quais, alguns são multifunções.
- 9 saídas e 6 entradas digitais.
- 3 saídas dreno aberto ou saídas espelho de corrente, programáveis de 0 a 20 mA.
- 4 canais de entrada analógicos.
- Reset externo configurável.
- Interface de programação de dois fios.

Interrupções

- Interrupções internas são para o conversor analógico para digital e para o *Timer*.
- Interrupções externas, podem ser configuradas com sensibilidade à borda ou nível.

Sumário

3 Designação dos pinos

Figura 1: Designação dos pinos

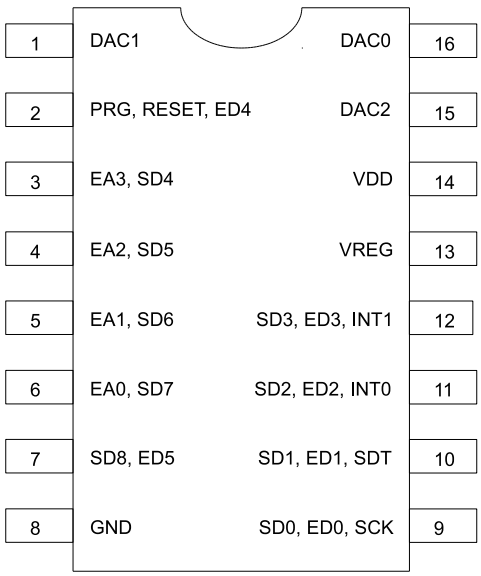


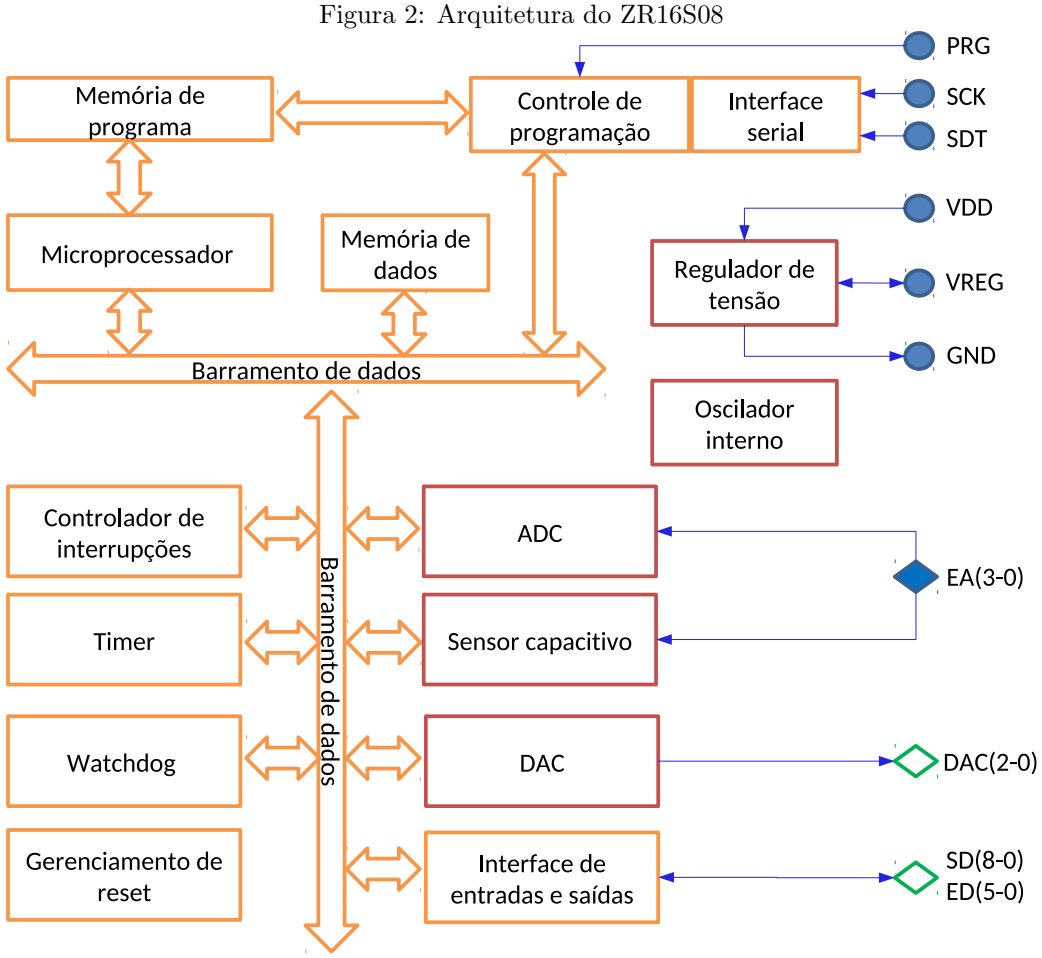
Tabela 1: Pinos do ZR16S08

Nome do sinal	Número do pino	Função	Descrição
DAC1	1	Saída	Saída dreno aberto/espelho de corrente 1
PRG	2	Entrada	Nível de tensão de programação
RESET		Entrada	Reset externo
ED4		Entrada	Entrada Digital 4
EA3	3	Entrada	Entrada Analógica 3
SD4		Saída	Saída Digital 4
EA2	4	Entrada	Entrada Analógica 2
SD5		Saída	Saída Digital 5
EA1	5	Entrada	Entrada Analógica 1
SD6		Saída	Saída Digital 6
EA0	6	Entrada	Entrada Analógica 0
SD7		Saída	Saída Digital 7
SD8	7	Saída	Saída Digital 8 ou saída do Sensor Capacitivo
ED5		Entrada	Entrada Digital 5
GND	8	Alimentação	Tensão de alimentação referência
SD0	9	Saída	Saída Digital 0
ED0		Entrada	Entrada Digital 0
SCK		Entrada	Sinal serial de sincronismo (programação)
SD1	10	Saída	Saída Digital 1
ED1		Entrada	Entrada Digital 1
SDT		Entrada	Sinal serial de dados (programação)
SD2	11	Saída	Saída Digital 2
ED2		Entrada	Entrada Digital 2
INT0		Entrada	Interrupção externa 0
SD3	12	Saída	Saída Digital 3
ED3		Entrada	Entrada Digital 3
INT1		Entrada	Interrupção externa 1
VREG	13	Alimentação	Saída do regulador de tensão de 3,3V
VDD	14	Alimentação	Tensão de alimentação
DAC2	15	Saída	Saída dreno aberto/espelho de corrente 2
DAC0	16	Saída	Saída dreno aberto/espelho de corrente 0

4 Arquitetura

O ZR16S08 conta com um microprocessador de 8 bits, RISC usando arquitetura Harvard, com 24 instruções. A memória de programa é do tipo EEPROM e a memória de dados SRAM. O microprocessador possui banco de 16 registradores sendo 13 de uso geral. Os outros 3 registradores contêm o *program counter*(PC, endereço atual de execução do programa), os controles e os *flags*. Possui um stack de 4 posições para armazenar flags, controles e PC. O microprocessador pode ser interrompido pela sinalização gerada a partir de 4 fontes configuráveis (ver em 8.1). Existem 9 saídas digitais, 6 entradas digitais (ver em 8.3), 3 saídas dreno aberto ou espelho de corrente(ver em 8.3.1) e 4 canais de entradas analógicas (ver em 8.2). Possui um reset externo que pode ser desabilitado. O mesmo pino do reset é utilizado para detectar nível de tensão em modo de programação do microcontrolador ou modo de execução. No modo de programação o microcontrolador pode receber dados para gravação ou leitura da EEPROM. Na Figura 4 é mostrada a arquitetura do ZR16S08 com os seus blocos e suas interconexões.

Figura 2: Arquitetura do ZR16S08



5 Espaços de Endereçamento

5.1 Banco de registradores

O ZR16S08 contém 16 registradores, entre os quais 13 são de uso geral. Estes registradores podem ser endereçados nas instruções diretamente por modos específicos de endereçamento. A organização dos registradores é mostrada na Tabela 2.

Registrador r15:

- Z: flag do sistema: Z=1 se o resultado da operação for zero.
- C: flag do sistema: informa se houve um *Carry* ou *Borrow* nas operações aritméticas (instruções ADD, SUB e COMP). Para as instruções ROT, SHL e SHA é o bit final resultado do deslocamento.
- V_P: flag do sistema: V_P= 1 quando o resultado de uma operação aritmética (ADD, SUB OU CMP) não puder ser representado em complemento de 2, ou quando o número de bits em 1 do resultado de uma operação lógica (AND, OR ou XOR) for par.
- uf1 a uf0: flags do usuário.
- uc: controle: quando uc = 1 o *carry* é usado nas instruções ADD, SUB, CMP e ROT.
- e/d: controle: usado nas instruções de ROT, SHL e SHA para informar se o deslocamento dos bits é para esquerda (e/d=1) ou direita (e/d=0).
- hint: controle: quando hint = 1 interrupções serão aceitas, senão as interrupções são ignoradas.

Registrador r14:

- mp: informa, quando estão sendo usado as instruções MOV rd, (ro) ou MOV r0, (end), se os dados são oriundos da memória de dados (mp=0) ou da memória de programa (mp=1).
- p2 a p0: quando mp=1, aponta a partição da memória de programa a ser acessada. São 8 partições de 128 x 16 bits.
- ppc9 e ppc8: são os próximos valores dos bits pc8 e pc9, se for usada uma instrução que altera o Program Counter (quando rd = r13), carregando diretamente o valor de um regis-

trador, ou usando o dado apontado por um registrador.

Registrador r13:

- pc7 a pc0: Bits 7 a 0 do *Program Counter*.

Tabela 2: Registradores do ZR16S08

Registrador	**Bit 7**	**Bit 6**	**Bit 5**	**Bit 4**	**Bit 3**	**Bit 2**	**Bit 1**	**Bit 0**
r15	Z	C	V_P	uf1	uf0	uc	e/d	hint
r14	mp	p2	p1	p0	ppc9	ppc8	pc 9	pc8
r13	pc7	pc6	pc5	pc4	pc3	pc2	pc1	pc0
r12	Registradores de 8 bits de uso geral							
r11								
r10								
r9								
r8								
r7								
r6								
r5								
r4								
r3								
r2								
r1								
r0								

5.2 Pilha de contexto do programa

Além dos 16 registradores o ZR16S08 possui 4 posições de pilha de contexto de programa (*stack*), onde são armazenadas as informações de ambiente no instante de execução de uma exceção. São consideradas exceções o salto para uma sub-rotina, usando a instrução CALL ou uma interrupção. O retorno para o fluxo de execução principal se dá pela instrução RET e suas variações. São armazenados na pilha:

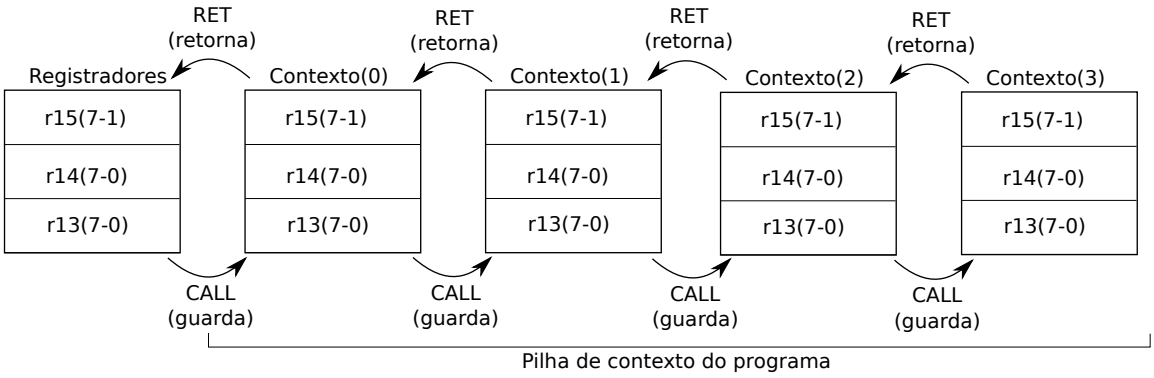
- registrador r13
- registrador r14
- registrador r15 - com excessão do bit hint

Quando ocorre uma exceção há um empilhamento, ou seja, o conteúdo dos registradores r15, r14 e r13 são armazenados no Contexto(0). O conteúdo de cada Contexto é salvo no Contexto de nível imediatamente superior. O conteúdo do Contexto(3) é perdido, por esta razão só é possível ter o contexto de 4 exceções empilhadas, sem que se execute um desempilhamento.

O desempilhamento dos contextos se dá na execução da instrução RET e de suas variações. Quando uma instrução RET é executada, o conteúdo é passado para os registradores r15, r14 e r13. Cada Contexto salva o conteúdo do Contexto imediatamente superior. O Contexto 3 tem o seu conteúdo zerado. Assim, existindo mais retornos de contexto (execução de RET) do que exceções o programa retornará à sua posição inicial (0x000)

A conteúdo da pilha só pode ser recuperado através da instrução RET e de suas variações. Só está disponível o contexto atual, ou seja o conteúdo dos registradores r15, r14 e r13.

Figura 3: Pilha de contexto do programa.



5.3 Endereçamento de periféricos

O espaço de endereçamento, reservado para os dados dos periféricos, é acessível através da especificação do tipo de acesso *IO* em uma instrução, como mostrado na instrução *mov io (r1),r0*. Para este tipo de acesso existem apenas 4 modos de endereçamento, conforme mostrado na seção 6.

Cada periférico tem registradores endereçáveis que podem ter diferentes conteúdos/significados. Os acessos a estes registradores são efetuados por operações de escrita ou leitura, conforme demonstrado na seção 8.

A Tabela 3 relaciona todos os periféricos e os respectivos endereços disponíveis no ZR16S08.

Tabela 3: Endereços periféricos ZR16S08

Endereço	**L/E**	**Periférico**	**Registrador**
0x00	E	Interrupções	Controle de Interrupções Externas
0x00	L	Interrupções	Status das Interrupções
0x04	E	ADC	Controle
0x04	L	ADC	Status & dados LSB
0x05	L	ADC	Dados MSB
0x08	L	Entradas e Saídas Digitais	Entradas digitais
0x09	L/E	Entradas e Saídas Digitais	Habilitação de saídas digitais
0x0A	L/E	Entradas e Saídas Digitais	Saídas Digitais
0x0B	L/E	Entradas e Saídas Digitais	Controle da saída SD8 e do reset externo
0x0C	L/E	Entradas e Saídas Digitais	Controle da saída DAC0
0x0D	L/E	Entradas e Saídas Digitais	Controle da saída DAC1
0x0E	L/E	Entradas e Saídas Digitais	Controle da saída DAC2
0x12	L/E	Sensor Capacitivo	Tensão Referência
0x13	L	Sensor Capacitivo	*Up Counter* - LSB
0x14	L	Sensor Capacitivo	Espelho do Contator - MSB
0x15	E	Sensor Capacitivo	Controle
0x15	L	Sensor Capacitivo	Status
0x16	E	Sensor Capacitivo	Set/Reset do Contador - LSB
0x17	E	Sensor Capacitivo	Set/Reset do Contador - MSB
0x1A	E	*Watchdog*	Controle
0x1A	L	*Watchdog*	Status
0x1C	E	*Timer*	Carga 0
0x1C	L	*Timer*	*Down Counter* - LSBs
0x1D	E	*Timer*	Carga 1
0x1D	L	*Timer*	Espelho do *Down Counter* - MSBs
0x1E	L	*Timer*	Status
0x1E	E	*Timer*	Controle
0x20	E	EEPROM	LSB palavra 0
0x21	E	EEPROM	MSB palavra 0
0x22	E	EEPROM	LSB palavra 1
0x23	E	EEPROM	MSB palavra 1
0x24	E	EEPROM	LSB palavra 2
0x25	E	EEPROM	MSB palavra 2
0x26	E	EEPROM	LSB palavra 3
0x27	E	EEPROM	MSB palavra 3
0x28	E	EEPROM	LSB palavra 4
0x29	E	EEPROM	MSB palavra 4
0x2A	E	EEPROM	LSB palavra 5
0x2B	E	EEPROM	MSB palavra 5
0x2C	E	EEPROM	LSB palavra 6
0x2D	E	EEPROM	MSB palavra 6
0x2E	E	EEPROM	LSB palavra 7
0x2F	E	EEPROM	MSB palavra 7
0x30	E	EEPROM	Endereço Apontador da Linha

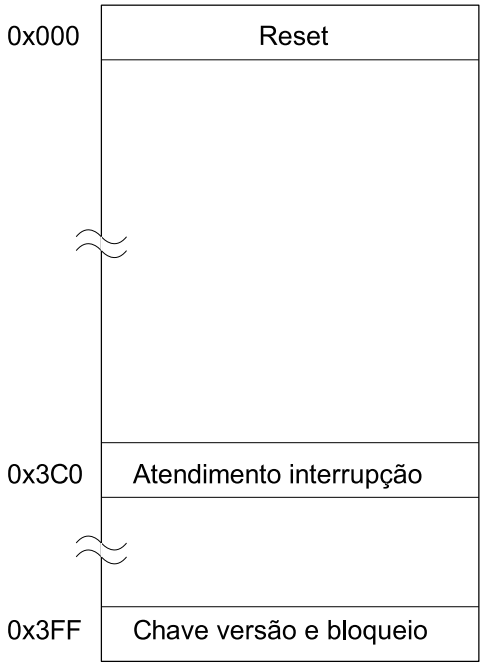
Convenção: L/E: acesso de leitura ou escrita; L : acesso de leitura; E : acesso de escrita

5.4 Memória de programa

O microcontrolador contém uma memória de programa interna, não volátil e reprogramável, uma EEPROM.

A capacidade da memória de programa do ZR16S08 é de 1024 palavras de 16 bits, conforme mostrado na Figura 18. Nesta Figura são destacados três endereços da EEPROM. O primeiro endereço é a posição 0x000 que é o ponto no qual o *Program Counter* inicia logo após o reset do ZR16S08. O endereço 0x3C0 é a posição para o atendimento de interrupções, endereço com o qual o *Program Counter* é carregado quando uma interrupção é atendida. O último endereço da EEPROM é a posição para gravação da versão do código e a chave para o bloqueio da memória de programa.

Figura 4: Memória de programa do ZR16S08



Acessos de leitura e escrita

Através do modo de programação é possível fazer acessos de leitura e escrita na EEPROM. Para maiores informações veja em 7.1.

Também é possível ler a memória de programa a partir de instruções do ZR16S08. Para fins de acesso como dados a memória de programa é particionada em 8 partes cada uma de 128 palavras de 16 bits. As instruções que acessam a memória do programa como dados são: mov r0,(end) e mov rd,(ro), sendo que para isto o bit mp de r14 tem que ser igual 1. A formação do endereço para a memória de programa está mostrado nas Figuras 5 e 6.

Para gravar a memória de programa a partir do ZR16S08 deve-se utilizar o periférico Controle de Programação, veja maiores detalhes em 8.7

Bloqueio da memória de programa

O último endereço da memória 0x3FF é reservado. Ele possui a chave de bloqueio da mémoria de programa no byte superior e a versão do programa no byte inferior, conforme descrito na Tabela 4.

Tabela 4: Codificação da última palavra da memória de programa.

Byte superior	Byte inferior
chave de bloqueio	número da versão do programa

Onde:

- chave de bloqueio: Informa se a memória está bloqueada.
  - chave de bloqueio = 0x5A, memória bloqueada
  - chave de bloqueio $\neq$ 0x5A memória desbloqueada
- número da versão do programa: número da versão do programa de 0x00 a 0xFF

Quando o ZR16S08 for inicializado em modo programação (ver informação sobre este modo de operação em 7.1), é realizada a leitura do endereço 0x3FF da memória de programa. Se chave de bloqueio for igual 0x5A só será possível a leitura desta posição de memória da EEPROM, protegendo assim o programa armazenado na EEPROM. Nesta situação. só é possível realizar leituras na posição do endereço 0x3FF ou apagar completamente a EEPROM para posteriormente reprogramá-la.

Durante o Modo de Execução esta palavra é lida como uma instrução qualquer, caso a execução atinja este endereço da memória.

Figura 5: Endereço da memória de programa para a intrução mov r0,(end) com mp=1

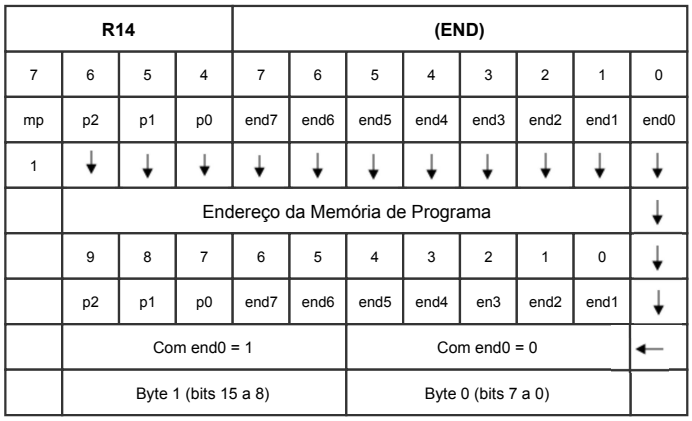
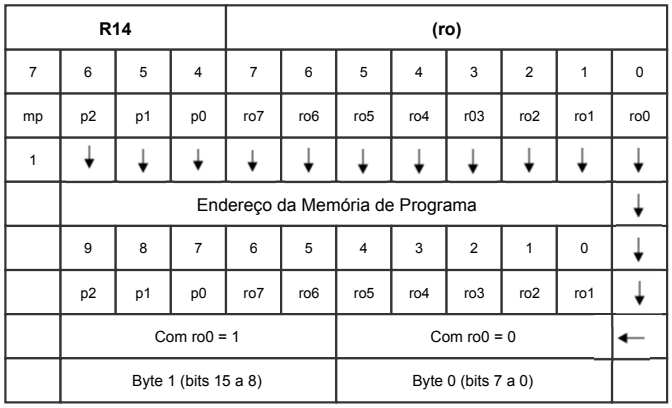


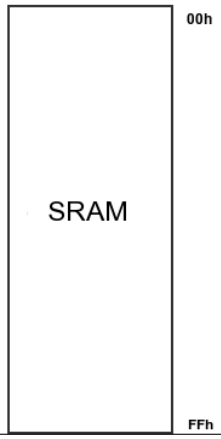
Figura 6: Endereço da memória de programa para a intrução mov rd,(ro) com mp=1



5.5 Memória de dados

A memória de dados é uma SRAM com capacidade de 256 x 8 bits (256 Bytes), é volátil, os dados são perdidos após um *reset* ou do desligamento do microcontrolador.

Figura 7: bloco da memória SRAM



Acessos à Memória de Dados O acesso ao barramento de dados da memória RAM, pode ser realizado de 5 maneiras.

- Acesso à memória de dados: leitura
- Acesso à memória de dados: escrita
- Acesso à memória de dados: leitura e escrita
- Acesso à memória de dados: duas leituras
- Acesso à memória de dados: duas leituras e uma escrita

6 Instruções e Endereçamentos

Intruções do ZR16S08 são apresentadas na Tabela 5.

Tabela 5: Instruções do ZR16S08

Instrução	Opcode(bits 15 a 12 da instrução)
JMP	0000
Jcc: JZ, JNZ, JC e JVP	0001
CALL	0010
RET, RETC, RETS, RETZ	0011 e bit 7 = 1
MVS	0011 e bit 7 = 0
AND	0100
OR	0101
XOR	0110
CMP	0111
ADD	1000
SUB	1001
ROT	1010
SHL	1011
SHA	1100
MOV	1101
DJNZ	1110
INC	1111 e bit 8 = 0
DEC	1111 e bit 8 = 1

As instruções do ZR16 tem um dos quatro formatos descritos a seguir:

INST MA DEST,ORIG: Este formato se aplica para as instruções: mov, cmp, add, sub, and, or , xor, rot, shl, sha, mvs.

INST MA DEST: Este formato se aplica para as instruções: inc,dec

INST DEST: Este formato se aplica para as instruções: jmp, call, jz, jnz, jc, jvp.

INST PA DEST: Este formato se aplica para as instruções: call, jmp, jz, jnz, jc, jvp

INST $reg_{1,2,3,4}$**,DEST**: Este formato se aplica para as instruções: djnz

INST: Este formato se aplica para as instruções: ret, rets, retz, retc

Onde:

INST: Instrução propriamente dita.

MA: Modificador de Acesso. O modificador é IO, e informa que o endereço da instrução é no espaço dos periféricos e não na memória de dados. Ele só está presente nos modos de endereçamento que fazem acessos no espaço de periféricos.

PA: Partição. Informa a partição da memoria de programa quando o endereço de salto vier de uma fonte que usa só 8 bits.

DEST,ORIG: São os operandos da instrução sendo DEST o operando destino e ORIG o operando origem.

$reg_{1,2,3,4}$: registradores usados pela instrução djnz.

As Tabelas 6 e 7 apresentam as combinações de operandos válidos para as instruções.

Exemplos:

mov io r0,(end)
mov é a instrução (movimentação), io informa que um dos operadores fica no espaço dos periféricos, r0,end são operadores sendo r0 o registrador de destino e (end) é endereço no espaço dos periféricos da origem.

add r1,(ro)
add é instrução (adição), r1,(ro): são os peradores. Sendo r1 o operando destino e (ro) o operando origem

shl r1,r2
shl é a instrução (*shift* lógico), r1,r2 são os operandos, sendo r1 operando destino e r2 o operando origem

jmp end_salto
jmp é a instrução e end_salto é valor absoluto do endereço de salto (destino).

djnz r3,end_salto
djnz é a instrução r3 o registrador que será decrementado e end_salto o endereço de salto enquanto r3 $\neq$ 0.

Tabela 6: Operando das instruções do ZR16S08

Instrução	**Descrição**	Notas
ro	registrador origem	
rd	registrador destino	
r0	registrador 0	1
$reg_{1,2,3,4}$	registrador 0	2
(ro)	posição de memória apontada pelo conteúdo do registrador origem	
(rd)	posição de memória apontada pelo conteúdo do registrador destino	
(r0)	posição de memória apontada pelo conteúdo do registrador 0	1
(end)	posição de memória apontada pelo valor de end	
imediato	valor de constante de 8 bits	
end10	valor de constante de 10 bits	3

Notas

1 Pode ser tanto como origem ou destino. Na instrução, quando é destino está à esquerda vírgula, quando é origem está à direita. Por exemplo, na instrução mov r0,imediato r0 é destino.

2 Registrador (r1,r2,r3 ou r4) que pode ser usado na instrução DJNZ.

3 Constante de 10 bits com o valor do endereço de salto, usado nas instruções JMP, Jcc, CALL e DJNZ.

Tabela 7: Instruções e Endereçamentos ZR16S08

Instrução	**Flag afetados**	**Ciclos para execução**	**Notas**
MOV rd,ro	Z	1	1
MOV r0,imediato	Z	1	
MOV rd,(ro)	Z	2	2, 3
MOV (rd),ro	Z	1	
MOV (rd),(ro)	Z	2	
MOV r0, (end)	Z	2	2
MOV (end), r0	Z	1	
MOV (r0), (end)	Z	2	
MOV (end), (r0)	Z	2	
MOV (r0), imediato	Z	1	
MOV io (end), r0	Z	1	4
MOV io r0, (end)	Z	2	4
MOV io (rd), ro	Z	1	4
MOV io rd, (ro)	Z	2	4, 3
MVS rd, imediato	Z	1	1
[ADD, SUB] rd, ro	Z, C, V_P	1	1
[ADD, SUB] rd, (ro)	Z, C, V_P	2	3
[ADD, SUB] r0, (end)	Z, C, V_P	2	
[ADD, SUB] r0, imediato	Z, C, V_P	1	
[ADD, SUB] (rd), ro	Z, C, V_P	2	
[ADD, SUB] (rd), (ro)	Z, C, V_P	3	
[ADD, SUB] (r0), (end)	Z, C, V_P	3	
[ADD, SUB] (r0), imediato	Z, C, V_P	2	
[ADD, SUB] (end), r0	Z, C, V_P	2	
[ADD, SUB] (end), (r0)	Z, C, V_P	3	
[ADD, SUB] io (end), r0	Z, C, V_P	2	4
[ADD, SUB] io r0, (end)	Z, C, V_P	2	4
[ADD, SUB] io (rd), ro	Z, C, V_P	2	4
[ADD, SUB] io rd, (ro)	Z, C, V_P	2	4, 3

Tabela 7: Instruções e Endereçamentos ZR16S08

Instrução	Flag afetados	Ciclos para execução	Notas
CMP rd,ro	Z, C, V_P	1	
CMP r0,imediato	Z, C, V_P	1	
CMP rd,(ro)	Z, C, V_P	2	
CMP r0,(end)	Z, C, V_P	2	
CMP (rd),ro	Z, C, V_P	2	
CMP (r0),imediato	Z, C, V_P	2	
CMP (end),r0	Z, C, V_P	2	
CMP (rd),(ro)	Z, C, V_P	3	
CMP (r0),(end	Z, C, V_P	3	
CMP (end),(r0)	Z, C, V_P	3	
CMP io (end),r0	Z, C, V_P	2	4
CMP io r0,(end)	Z, C, V_P	2	4
CMP io (rd),ro	Z, C, V_P	2	4
CMP io rd,(ro)	Z, C, V_P	2	4
INC rd	Z	1	1
DEC rd	Z	1	1
INC (rd)	Z	2	
DEC (rd)	Z	2	
INC (end)	Z	2	
DEC (end)	Z	2	
INC io (end)	Z	2	4
DEC io (end)	Z	2	4
INC io (rd)	Z	2	4
DEC io (rd)	Z	2	4
[AND, OR, XOR] rd, ro	Z, V_P	1	1
[AND, OR, XOR] rd, (ro)	Z, V_P	2	3
[AND, OR, XOR] r0, (end)	Z, V_P	2	
[AND, OR, XOR] r0, imediato	Z, V_P	1	
[AND, OR, XOR] (rd), ro	Z, V_P	2	
[AND, OR, XOR] (rd), (ro)	Z, V_P	3	
[AND, OR, XOR] (r0), (end)	Z, V_P	3	
[AND, OR, XOR] (r0), imediato	Z, V_P	2	
[AND, OR, XOR] (end), r0	Z, V_P	2	
[AND, OR, XOR] (end), (r0)	Z, V_P	3	
[AND, OR, XOR] io (end), r0	Z, V_P	2	4
[AND, OR, XOR] io r0, (end)	Z, V_P	2	4
[AND, OR, XOR] io (rd), ro	Z, V_P	2	4
[AND, OR, XOR] io rd, (ro)	Z, V_P	2	4, 3
ROT rd, ro	Z, C	1	1
ROT rd,(ro)	Z, C	2	3
ROT r0,(end)	Z, C	2	
ROT (rd),ro	Z, C	1	
ROT (rd),(ro)	Z, C	2	
ROT (r0),(end)	Z, C	2	
ROT (end),r0	Z, C	1	
ROT (end),(r0)	Z, C	2	
ROT io (end),r0	Z, C	1	4
ROT io r0,(end)	Z, C	2	4
ROT io (rd),ro	Z, C	1	4
ROT io rd,(ro)	Z, C	2	4, 3
[SHL, SHA] rd, ro	Z, C	1	1, 5
[SHL, SHA] rd, (ro)	Z, C	2	3, 5
[SHL, SHA] r0, (end)	Z, C	2	5
[SHL, SHA] (rd), ro	Z, C	1	5
[SHL, SHA] (rd), (ro)	Z, C	2	5
[SHL, SHA] (r0), (end)	Z, C	2	5
[SHL, SHA] (end), r0	Z, C	1	5
[SHL, SHA] (end), (r0)	Z, C	2	5
[SHL, SHA] io (end), r0	Z, C	1	4, 5
[SHL, SHA] io r0, (end)	Z, C	2	4, 5
[SHL, SHA] io (rd), ro	Z, C	1	4, 5
[SHL, SHA] io rd, (ro)	Z, C	2	4, 3, 5

Tabela 7: Instruções e Endereçamentos ZR16S08

Instrução	Flag afetados	Ciclos para execução	Notas
JMP par rd		2	
JMP par (rd)		3	
JMP end10		2	
JZ end10		1	7
JNZ end10		1	7
JC end10		1	7
JVP end10		1	7
CALL par rd		2	
CALL par (rd)		3	
CALL end10		2	
RET		2	
RETZ	hint	2	
RETS	hint	2	
RETC	hint	2	
DJNZ $reg_{1,2,3,4}$**, end10**	Z	2	6

Notas
1 Se registrador de destino for R13 a instrução será executada em 2 ciclos.
2 Se $r14_{mp} =' 1'$ **instrução buscará dado na EEPROM. Ex:Figuras 5 6**
3 Se registrador de destino for R13 a instrução será executada em 3 ciclos.
4 O modificador io informa que os endereços de dados são do espaço de endereçamentos dos periféricos.
5 Se a instrução for SHA a flag V_P também será afetada.
6 Se a condição de salto Z=0 não for satisfeita a instrução é executada em 1 ciclo.
7 Se a condição de salto (z, nz, c, v_p) for satisfeita a instrução será executada em 2 ciclos.
8 Se IO 0xFE$_{mp}$**=1 e ocorrer uma interrupção durante a execução da instrução, o conteúdo do flag Z armazenado em r15 é indeterminado.**

O ZR16S08 possibilita realizar operações com os registradores de controle R13, R14 e R15, e quando o destino dessas operações for os registradores de controle, elas terão um comportamento específico.

Caso destino da operação lógica seja r15 os flags Z, C e V_P serão carregadas com o valor do transferido para estes bits.

Se rd≠r15 então:
Z = 1 se o resultado da operação = x"00"
senão 0

C e V_P são alterados de acordo com a instrução.

Se o destino for r13 ou r14 a instrução tem um comportamento diferenciado que está detalhado no texto da execução da instrução.

Se rd = r14 então:
$r14_{7a2}$ = o resultado da operação$_{7a2}$
$r14_{1a0} = r14_{1a0}$

Se rd = r13 então:
r13 = resultado da operação
$r14_{1a0} = r14_{3a2}$
Ou seja: PC = ppc9 & ppc8 & resultado da operação
senão:
PC = PC+1

MOV

Na instrução MOV, o conteúdo é copiado do operando de origem para o destino. O valor da origem é mantido e o valor do destino é sobrescrito.

Quando $r14_7$=1 (mp) e for executada a instrução *MOV r0, (end)* ou *MOV rd, (ro)* é transferido um dado da memória de programa para o registrador especificado. A formação do endereço da memória de programa é feito conforme as Figuras 5, 6.

O flag Z é acertado de acordo com o resultado da movimentação. Os flags C e V_P não são afetados.

Nota: *quando ocorrer uma interrupção durante excução da instrução MOV r0,(end) ou da instrução MOV rd,(ro) e o bit mp=1, o conteúdo do flaz Z é indeterminado, não podendo ser usado para decisões de mudança de fluxo, como por exemplo da instrução jnz end ou da instrução jz end. Ver também nota 8 na tabela .7*

MVS

A instrução MVS carrega em registrador (r0 a r15) com o valor imediato contido nos bits 6 a 0 da instrução (valor entre: 0 a 7F).

O flag Z é carregado de acordo com o resultado da transferência. Os flags C e V_P não são afetados.

Caso o destino da operação lógica seja r15, neste caso Z será sempre carregado com '0'.

ADD, SUB

A instrução [ADD, SUB] efetua a adição (ADD) ou subtração (SUB) de dois operandos. O resultado é carregado no operando destino.

ADD:
DEST = DEST + ORIG + (C da instrução anterior se $r15_2$=1, senão 0)
SUB:

DEST = DEST - ORIG - (C da instrução anterior se $r15_2$=1, senão 0)

Comentário: Regra para obtenção do vetor de carry da ALU: C_8 a C_0

Se $r15_2$=1 (uc=1) então:
C_0 = C resultante da instrução anterior
senão:
C_0 = 0 para i=0,1 ...7
Se instrução = ADD então:
Ci+1 = ($DEST_i$ and $ORIG_i$) or ($ORIG_i$ and C_i) or ($DEST_i$ and C_i)
Se instrução = SUB então:
C_i+1 = ((not $DEST_i$) and C_i) or ((not $DEST_i$) and $ORIG_i$) or ($ORIG_i$ and C_i)
C = C_8
V_P = C_8 xor C_7

Onde "DEST" e "ORIG" são de acordo com o endereçamento da instrução na tabela 7.

CMP

A instrução CMP executa a operação de subtração do operando destino pelo operando origem. Conforme descrito abaixo.

[DEST - ORIG - (C da instrução anterior se $r15_2$ = 1, senão 0)]

Comentário: Regra para obtenção do vetor de carry da ALU: C8 a C0

Se $r15_2$ = 1 (uc=1) então:
C_0 = C resultante da instrução anterior
senão:
C_0 = 0
para i = 0,1 ... 7
C_{i+1} = ((not $DEST_i$) and C_i) or ((not $DEST_i$) and $ORIG_i$) or ($ORIG_i$ and C_i)
C = C_8
V_P = C_8 xor C_7

Onde "DEST" e "ORIG" são de acordo com o endereçamento da instrução na tabela 7.

Na instrução CMP o destino fica inalterado, só são acertados os *flags* Z, C e V_P de acordo com o resultado da operação.

INC, DEC

A instrução [INC, DEC] incrementa (INC) ou decrementa (DEC) o valor do operando que pode ser um registrador, valor do dado da memória de dados ou do IO. Quando for um dado da memória ou de IO, o seu endereço pode ser fornecido diretamente na instrução ou indiretamente por um registrador.

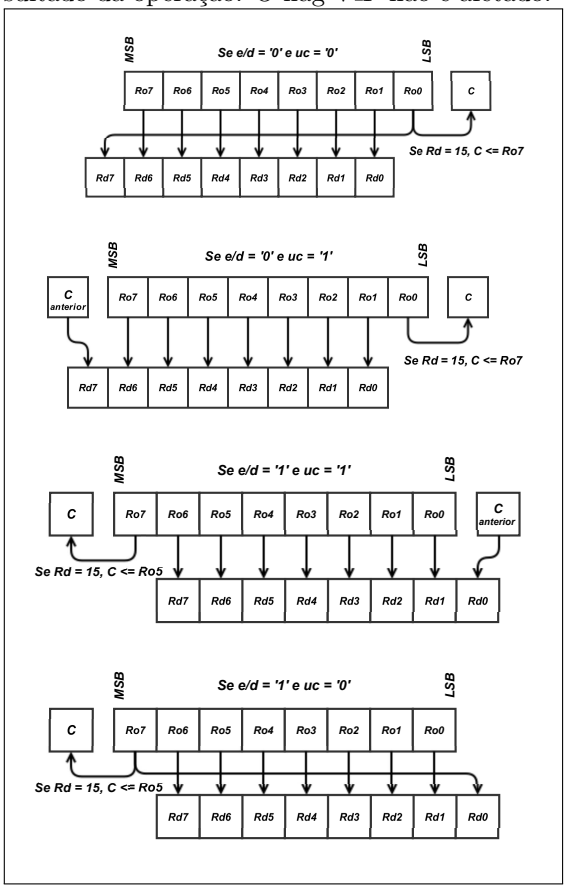
O flag Z é acertado de acordo com o resultado da movimentação. Os flags C e V_P não são afetados.

AND, OR, XOR

A instrução [AND, OR, XOR] efetua a operação lógica [AND, OR, XOR], bit a bit com o operando destino e o operando origem. O resultado é carregado no operando destino. O flag Z é acertado de acordo com o resultado da operação. O flag C não é afetado pela operação. O flag V_P é carregado de acordo com a paridade do resultado, '1' se o numero de bits em '1' do resultado for par, senão '0'.

ROT

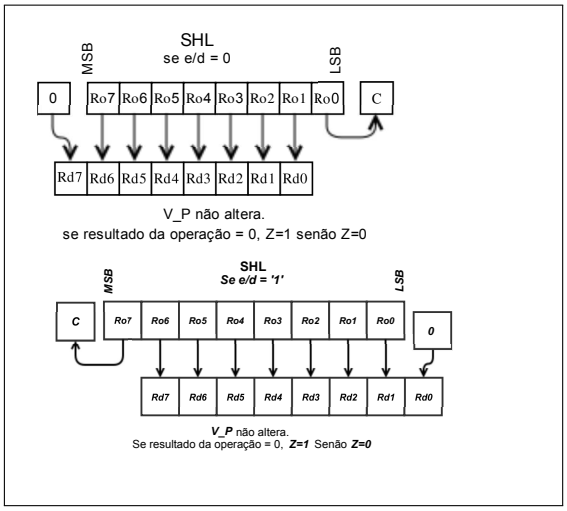
A instrução ROT faz a operação de rotação lógica de uma posição do dado de origem, pondo o resultado da rotação no destino. A rotação para esquerda ou para direita é definido por $r15_1$(e/d). Se $r15_1$= 1 roda para esquerda, senão direita. O uso do flag C na rotação é definido por $r15_2$(uc). Se $r15_2$=1 usa C senão não usa. Os flags Z e C são acertados de acordo com o resultado da operação. O flag V_P não é afetado.

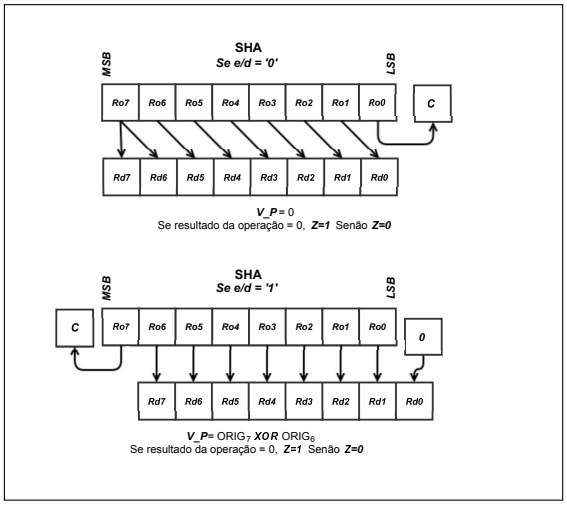


SHL, SHA

A instrução [SHL, SHA], executa o deslocamento lógico [SHL] ou aritmético [SHA] de um bit para a esquerda ou para a direita, do dado origem, pondo o resultado no destino. O deslocamento para esquerda ou para direita é definido por $r15_1$(e/d). Se $r15_1$= 1 desloca para esquerda, senão direita.

Os flags Z e C são acertados de acordo com o resultado da operação. Quando a instrução for SHA o bit V_P também tem seu valor definido de acordo com o resultado da operação.





JMP

A instrução JMP permite ao microprocessador saltar a execução do programa para um endereço determinado.

O endereço de destino pode ser o conteúdo de um registrador, o conteúdo do endereço da memória de dados apontado por um registrador ou um endereço absoluto fornecido na própria palavra da instrução.

No caso de o endereço destino ser o conteúdo do endereço da memória de dados apontado por um registrador é feito um acesso de leitura na memória de dados para obter o conteúdo do endereço apontado pelo registrador.

Quando o endereço é fornecido pelo conteúdo de um registrador ou pelo conteúdo do endereço de memória de dados apontado por um registrador, a memória de programa é considerada como se tivesse 4 partições de 256 endereços. Os bits que informam a partição (p1 e p0) são fornecidos na instrução e são usados como bits 9 e 8 no PC.

JCC: JZ, JNZ, JC, JVP

Quando da execução da instrução Jcc o microprocessador salta para um endereço determinado se o teste do flag especificado for verdadeiro. O endereço de salto é fornecido na própria instrução.

- JZ salta quando $Z = 1$
- JNZ salta quando $Z = 0$
- JC salta quando $C = 1$
- JVP salta quando $V_P = 1$

CALL

A instrução CALL possibilita ao microprocessador saltar a execução do programa para um endereço determinado, salvando na pilha de contexto o ambiente de execução em que estava (PC, controles e flags), conforme a figura 3.

O endereço de destino pode ser o conteúdo de um registrador, o conteúdo do endereço da memória de dados apontado por um registrador ou um endereço absoluto fornecido na própria palavra da instrução.

Quando o endereço destino é o conteúdo do endereço da memória de dados apontado por um registrador, é feito um acesso de leitura na memória de dados para obter o conteúdo do endereço apontado pelo registrador.

Quando o endereço é fornecido pelo conteúdo de um registrador ou pelo conteúdo do endereço de memória de dados apontado por um registrador, a memória de programa é considerada como se tivesse 4 partições de 256 endereços. Os bits que informam a partição (p1 e p0) são fornecidos na instrução e são usados como bits 9 e 8 no PC.

Os flags Z, C e V_P não são afetados quando da execução desta instrução.

RET

A instrução RET é usada para retornar o ambiente original de execução (PC, controles e *flags*). Ela é usada tanto no caso de interrupção, como para retornar de chamada de sub-rotina (CALL).

A instrução RET possibilita a programação do bit hint ($r15_0$). $r15_0$ pode ser mantido como está (RET), zerado (RETZ), ligado (RETS) ou complementado (RETC). Durante a execução da instrução RET, mesmo que haja uma interrupção e hint = 1 (habilitação de interrupção), não ocorre o aceite de interrupção.

DJNZ

A instrução DJNZ decrementa um registrador especificado (r1, r2, r3 ou r4) e verifica se o resultado é zero. Se o resultado for zero, executa a próxima instrução, senão salta para o endereço especificado na instrução.

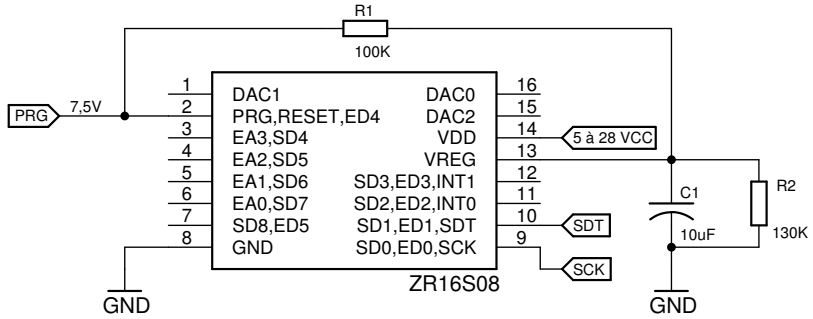
O flag Z acertado de acordo com o resultado do decremento do registrador. Os flags C e V_P permanecem inalterados.

7 Modos de operação

7.1 Modo Programação

O modo de programação é ativado quando a tensão no pino PRG/RST/ED4 ≈ 7,5V. A tensão neste pino não deve ultrapassar 10V. Quando em modo de programação, os pinos SD1/ED1/SDT e SD0/ED0/SCK passam a funcionar como interface serial de dois fios, usada na programação, como mostrado na configuração da Figura 8.

Figura 8: Configuração recomendável para entrada em Modo programação



7.2 Modo Execução e *Reset*

7.2.1 Modo Execução

Quando a tensão no pino PRG/RST/ED4 é de ≈ 3,3V ocorre a inicialização do microprocessador, este passa a efetuar leitura da memória de programa para obter intruções e estas são executadas. A configuração recomendável para operação neste modo é apresentada na Figura 9.

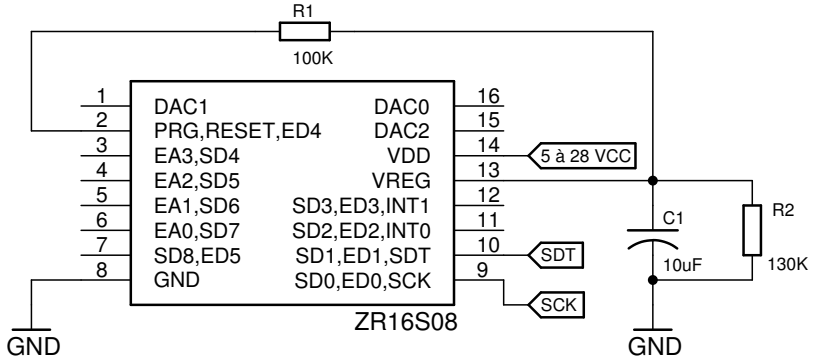
7.2.2 *Reset*

O *reset* do microcontrolador faz com que ele seja configurado para uma condição pré-determinada, independentemente de parâmetros externos. Após o *reset* o microprocessador buscará a primeira instrução a ser executada no endereço 0x000 da memória de programa.

O ZR16S08 possui três fontes possíveis de *reset*:

- *Power On Reset* (POR) do microcontrolador: O POR é um circuito de controle, interno, que monitora o regulador de tensão e o oscilador internos do ZR16S08. Ao ligar, ele mantém o microcontrolador em *reset*, enquanto o regulador de tensão e o oscilador não estiverem funcionando.
- *Watchdog*: temporizador que, quando habilitado, ativa o reset do ZR16S08 após ter transcorrido o tempo programado em seu Registrador de Controle, maiores detalhes ver no item 8.5.
- *Reset* externo: Pode ser aplicado um *reset* no ZR16S08 fornecendo uma tensão de ≈ 0V no pino PRG/RST/ED4. Para que isto ocorra também é necessário que o **bit cre** do **Registrador de Contrlole do SD8 e Interrupção Externas** seja igual a 0, habilitando o reset externo. Ver item 8.3 para maiores detalhes.

Figura 9: Configuração recomendável para entrada em Modo execução



8 Periféricos

8.1 Interrupção

Esse bloco é responsável pelo controle de interrupções internas e externas do ZR16S08.

Para ocorrer uma interrupção é necessário que o bit hint do registrador r15 do Banco de Registradores esteja ativado, se a instrução *RET* ou *(RET, RETC, RETZ ou RETS)*, estiver sendo executada no momento de uma interrupção o microprocessador só irá considerá-la na próxima instrução que não seja do tipo *RET*. Para maiores informações sobre o registrador r15 veja 5.1.

Uma vez satisfeitas as condições expostas anteriormente, os seguintes eventos ocorrem:

- A instrução que seria carregada no registrador de instruções é substituída por JMP 0x3C0
- $r15_{7a1}$, r14, r13 são armazenados na pilha de contexto de programa
- $r15_0$(hint) = '0'
- $r14_7$(mp) = '0'

As Figuras 10, 11 e 12 demonstram como ocorre a interrupção para instruções executadas em um, dois ou três ciclos de *clock*, respectivamente.

Na borda de subida do *clock* do fim do ciclo de execução da instrução i, a instrução que normalmente seria armazenada seria a instrução i+1. No entanto, como o sinal interrupção está ativo (ativo em nível baixo, interrupção=0) e hint=1, considerando que a instrução i não é um RET, o Microprocessador aceita a interrupção, pondo no lugar da instrução i+1, o JMP 0x3C0, carregando o *stack* com r15(bits 7 a 1), r14 e r13, e zerando os bits hint e mp. Dessa forma o endereço "end i+1"é salvo no *stack* e a instrução JMP 0x3C0 é executada como se ela tivesse sido buscada da EEPROM.

As fontes internas de interrupção são o ADC e o *Timer*.

Como fonte externa existem dois sinais de entrada que podem ser habilitados a interromper. Através do Registrador de Controle de Interrupção Externa é possível habilitar a interrupção e escolher a causa da interrupção (borda de subida, borda de descida, nível zero, nível um) destes sinais.

> **Nota:** *quando ocorrer uma interrupção durante excução da instrução MOV r0,(end) ou da instrução MOV rd,(ro) e o bit mp=1, o conteúdo do flaz Z é indeterminado, não podendo ser usado para decisões de mudança de fluxo, como por exemplo da instrução jnz end ou da instrução jz end. Ver também nota 8 na tabela .7*

Registradores utilizados:

- Registrador de Controle de Interrupção Externa: **0x00** (escrita)
- Registrador de Status das Interrupções: **0x00** (leitura)

Registrador de Controle das Interrupções Externas

	Endereço: 0x00							
	Tipo: escrita							
Índice	7	6	5	4	3	2	1	0
Bit	IC1(3)	IC1(2)	IC1(1)	IC1(0)	IC0(3)	IC0(2)	IC0(1)	IC0(0)
Reset	0	0	0	0	0	0	0	0

A Tabela 8 apresenta os possíveis valores para IC1 e IC0 do Registrador de Controle das Interrupções Externas.

Onde:

- Os bits IC1(3) IC0(3) não são memorizados. Eles limpam a interrupção correspondente se estiverem em '1' quando da escrita no registrador.

Registrador de Status das Interrupções

	Endereço: 0x00							
	Tipo: leitura							
Índice	7	6	5	4	3	2	1	0
Bit	x				int_timer	int_adc	int_exte1	int_exte0
Reset	x				0	0	0	0

Onde:

- x: *don't care*
- int_ext0
  - 1: interrupção externa 0
  - 0: não há interrupção externa 0
- int_ext1
  - 1: interrupção externa 1
  - 0: não há interrupção externa 1
- int_adc
  - 1: interrupção do ADC
  - 0: não há interrupção do ADC
- int_timer
  - 1: interrupção do timer
  - 0: não há interrupção do timer

O Registrador de Status das Interrupções contém a(s) fonte(s) de interrupção que acionaram o sinal de interrupção do ZR16S08.

Para as *flags* de *status* serem limpas, é necessário fazer leitura no registrador de status do bloco que originou a interrupção, *Timer*, já para o ADCler em 8.2.

Figura 10: Atendimento de interrupção para instruções executadas em apenas um ciclo de *clock*.

Figura 11: Atendimento de interrupção para instruções executadas em dois ciclos de *clock*.

Figura 12: Atendimento de interrupção para instruções executadas em três ciclos de *clock*.

Tabela 8: Possíveis valores para IC1 e IC0 do Registrador de Controle das Interrupções Externas.

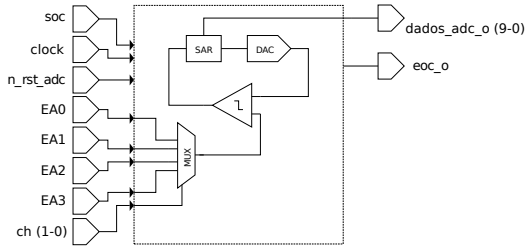
ICn(3)	ICn(2)	ICn(1)	ICn(0)	n = 0, 1
0	x	x	x	Mantém a interrupção n
1	x	x	x	Limpa a interrupção n
x	0	x	x	Desabilita a interrupção n
x	1	x	x	Habilita a interrupção n
x	x	0	0	Interrupção n ocorre no nível zero
x	x	0	1	Interrupção n ocorre na borda de subida
x	x	1	0	Interrupção n ocorre na borda de descida
x	x	1	1	Interrupção n ocorre no nível um

8.2 ADC

Este periférico é um conversor analógico-digital de 10 bits, que utiliza o princípio de *Successive Approximation Register* (SAR). O diagrama de blocos juntamente com as 4 entradas que são multiplexadas para o periférico, EA0, EA1, EA2 e EA3, pode ser visualizado na Figura 13.

O ADC opera de modo linear e utiliza como referência positiva a tensão de saída do regulador de tensão 3.3V e como referência inferior o GND.

Figura 13: Diagrama de blocos do ADC



Registradores utilizados:

- Registrador de Controle: **0x04** (escrita)
- Registrador de Status & dados LSB: **0x04** (leitura)
- Registrador de Dados MSB: **0x05** (leitura)

Registrador de Controle

	Endereço: 0x04							
	Tipo: escrita							
Índice	7	6	5	4	3	2	1	0
Bit	x	x	soc	hb_int	x	n_reset_adc	ch1	ch0
Reset	0	0	0	0	0	0	0	0

Onde:

- x: *"Don't Care"*
- soc: *Start of Conversion*, bit utilizado para sinalização do início da conversão do ADC.
- hb_int: controla interrupção da conversão do ADC
  - 1: interrupção habilitada ao fim da conversão
  - 0: interrupção desabilitada
- n_reset_adc: controla reset do periférico
  - 1: não realiza reset do periférico
  - 0: ativa reset do periférico
- ch1,ch0: Canal analógico a ser convertido
  - 0, 0: EA0
  - 0, 1: EA1
  - 1, 0: EA2
  - 1, 1: EA3

A operação do periférico é iniciada com a escrita de uma palavra no Registrador de Controle com n_reset_adc habilitado e soc habilitado, na saída da condição de *reset* do sistema o conversor já se encontra pronto para operação. Se deseja-se interromper a operação (depois de iniciada a conversão) pode-se escrever o Registrador de Controle com n_reset_adc habilitado, parando assim a conversão e deixando o conversor em um estado pronto para iniciar uma nova conversão ao comando do usuário.

Para limpar a interrupção é necessário resetar o bloco ADC, através da escrita do bit n_reset_adc. Mas também é possível desligar a interrupção atual só dando um novo SOC ou desabilitando a interrupção do bloco. O mesmo vale para limpar o bit eoc do Registrador de Status & Dados LSB.

Registrador de Status & dados LSB

	Endereço: 0x04							
	Tipo: leitura							
Índice	7	6	5	4	3	2	1	0
Bit	x					eoc	adc1	adc0
Reset	x					0	0	0

Onde:

- x: *"Don't Care"*
- eoc: *End Of Conversion*
  - 1: Conversão finalizada, dados disponiveís
  - 0: Conversão em andamento ou conversor parado
- adc1: Bit 1 da palavra do resultado da conversão
- adc0: Bit 0 da palavra do resultado da conversão

O fim da conversão pode ser observado pelo usuário através da geração de uma interrupção ou por meio da leitura do bit eoc. Após encerrada uma conversão e imediatamente depois de disparada uma nova conversão o bit eoc permanece ativo por poucos ciclos de clock do sistema ainda indicando status da conversão anterior antes de ter nível lógico trocado para baixo.

Registrador de Dados MSB

	Endereço: 0x05							
	Tipo: leitura							
Índice	7	6	5	4	3	2	1	0
Bit	adc9	adc8	adc7	adc6	adc5	adc4	adc3	adc2
Reset	0	0	0	0	0	0	0	0

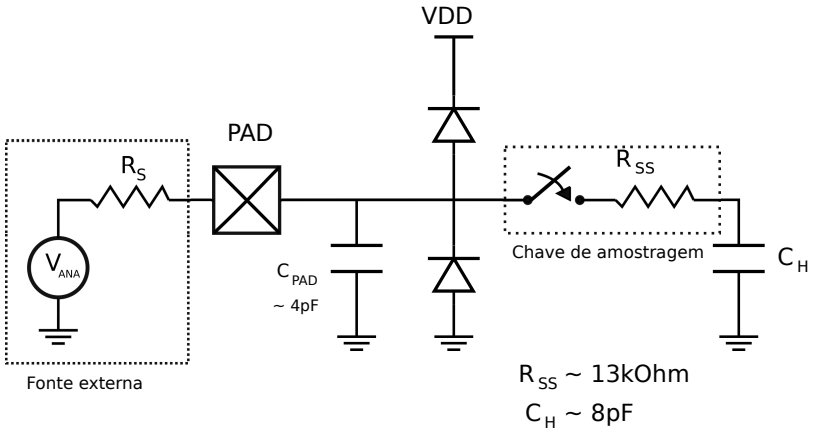
Onde:

- adc (9-2): Bits correspondentes da palavra do resultado da conversão

Requisitos para aquisição

Mais características elétricas do conversor podem ser encontradas no Capítulo 9. Para a conversão do ADC atingir os valores especificados na Seção 9 é importante considerar o modelo da entrada analógica apresentado na Figura 14.

Figura 14: Modelo de entrada analógica do ADC



8.3 Entradas e Saídas Discretas

Este periférico permite a habilitação e o acionamento das 9 saídas digitais e a leitura de 6 entradas digitais. Nele é possível programar à habilitação do uso do mesmo pino de *reset* externo (PRG/RESET/ED4) como entrada digital comum, desabilitando o seu uso como *reset*.

Nele também se encontra o controle dos 3 DACs do ZR16S08.

Registradores utilizados:

- Registrador de entrada digital: **0x08** (leitura)
- Registrador de habilitação de saída digital: **0x09** (leitura/escrita)
- Registrador de saídas digitais: **0x0A** (leitura/escrita)
- Registrador de controle SD8: **0x0B** (leitura/escrita)

Registrador de leitura de entradas digitais

	Endereço: 0x08							
	Tipo: leitura							
Índice	7	6	5	4	3	2	1	0
Bit	0	0	ed5	ed4	ed3	ed2	ed1	ed0
Reset	0	0	0	0	0	0	0	0

Onde:

- ed[5:0]:bits com nível lógico da entrada digital.

Este registrador armazena os sinais das entradas digitais de 0 à 5. Eles estão disponíveis para a leitura pelo microprocessador no barramento de dados. Estes valores são válidos para leitura ainda que a função dos pinos equivalentes seja compartilhada com outras funções como saídas digitais.

Registrador habilitação de saídas digitais

	Endereço: 0x09							
	Tipo: leitura/escrita							
Índice	7	6	5	4	3	2	1	0
Bit	hsd7	hsd6	hsd5	hsd4	hsd3	hsd2	hsd1	hsd0
Reset	0	0	0	0	0	0	0	0

Onde:

- hsd[7:0]:bits que habilitam saída digital correspondente quando em nível lógico alto.

Registrador de controle de saídas digitais

	Endereço: 0x0A							
	Tipo: leitura/escrita							
Índice	7	6	5	4	3	2	1	0
Bit	rsd7	rsd6	rsd5	rsd4	rsd3	rsd2	rsd1	rsd0
Reset	0	0	0	0	0	0	0	0

Onde:

- rhsd[7:0]:bits referente ao controle do nível lógico da saída digital correspondente.

Registrador de controle de SD8 e reset externo

	Endereço: 0x0B							
	Tipo: leitura/escrita							
Índice	7	6	5	4	3	2	1	0
Bit	cre	x	x	x	h_sd8	t_sinc	sai_comp	rsd8
Reset	0	0	0	0	0	0	0	0

Onde:

- cre: Controle de reset externo
  - 0: Habilita *reset* externo, fazendo com que ED4 seja usado como valor deste *reset*.
  - 1: Desabilita o *reset* externo, fazendo com que ED4 possa funcionar como uma entrada digital comum.
- h_sd8: Habilita saída digital 8
  - 0: Saída desabilitada
  - 1: Habilita saída.
- t_sinc: Escolha da fonte para gerar r_sd8 a partir do sensor capacitivo (habilitada por sai_comp) (ver 8.4)
  - 0: Usa saída do comparador do sensor capacitivo (ff_cmp) para gerar SD8
  - 1: Usa saída do comparador do sensor capacitivo sincronizada no *clock* do microprocessador (4MHz) para gerar SD8
- sai_comp: Habilita uso da saída do sensor capacitivo com a saída digital SD8 (ver 8.4)
  - 0: usa saída do digital 8 com o valor do bit rsd8
  - 1: usa saída SD8 de acordo com configuração de t_sinc
- rsd8: Bit com nível lógico de SD8 quando sai_comp = 0 e h_sd8 = 1

8.3.1 DACs

Os DACs consitem em 3 saídas de corrente controladas digitalmente, com funcionamente indepentete. Cada uma das 3 saídas tem um transistores NMOS do tipo dreno aberto com capacidade de 30V/20mA. Estes transistores podem funcionar como acionadores de LEDs ou como chaves de potência para o acionamento de relés.

Registradores utilizados:

- Registrador de controle da saída DAC0: **0x0C** (leitura/escrita)
- Registrador de controle da saída DAC1: **0x0D** (leitura/escrita)
- Registrador de controle da saída DAC2: **0x0E** (leitura/escrita)

Registrador de controle da saída DAC0

	Endereço: 0x0C							
	Tipo: leitura/escrita							
Índice	7	6	5	4	3	2	1	0
Bit	mod_c(0)	0/x	0/x	dac(0)	set_c0(3)	set_c0(2)	set_c0(1)	set_c0(0)
Reset	0	0	0	0	0	0	0	0

Onde:

- mod_c(0): modo de operação da saída DAC0 (0: chave liga/desliga, 1: espelho de corrente);
- dac(0): bit para acionamento da saída dreno aberto 0 quando atuando como chave liga/-desliga;
- set_c0[3:0]: valor da corrente do dac0, no modo espelho de corrente. Pode variar de 0 a 20mA com 16 palavras de configuração com incrementos de 1,33 mA;
- 0/x: leitura: 0, escrita: *don't care.*

Registrador de controle da saída DAC1

	Endereço: 0x0D							
	Tipo: leitura/escrita							
Índice	7	6	5	4	3	2	1	0
Bit	mod_c(1)	0/x	0/x	dac(1)	set_c1(3)	set_c1(2)	set_c1(1)	set_c1(0)
Reset	0	0	0	0	0	0	0	0

Onde:

- mod_c(1): modo de operação da saída DAC1 (0: chave liga/desliga, 1: espelho de corrente);
- dac(1): bit para acionamento da saída dreno aberto 1 quando atuando como chave liga/-desliga;
- set_c1[3:0]: valor da corrente do dac1, no modo espelho de corrente. Pode variar de 0 a 20mA com 16 palavras de configuração com incrementos de 1,33 mA;
- 0/x: leitura: 0, escrita: *don't care.*

Registrador de controle da saída DAC2

	Endereço: 0x0E							
	Tipo: leitura/escrita							
Índice	7	6	5	4	3	2	1	0
Bit	mod_c(2)	0/x	0/x	dac(2)	set_c2(3)	set_c2(2)	set_c2(1)	set_c2(0)
Reset	0	0	0	0	0	0	0	0

Onde:

- mod_c(2): modo de operação da saída DAC2 (0: chave liga/desliga, 1: espelho de corrente);
- dac(2): bit para acionamento da saída dreno aberto 2 quando atuando como chave liga/-desliga;
- set_c2[3:0]: valor da corrente do dac2, no modo espelho de corrente. Pode variar de 0 a 20mA com 16 palavras de configuração com incrementos de 1,33 mA;
- 0/x: leitura: 0, escrita: *don't care.*

8.4 Sensor Capacitivo

O Sensor Capacitivo é composto por dois blocos um analógico e outro digital. O bloco analógico processa as entradas analógica convertendo os resultados para um nível digital ou para um pulso digital, de acordo com sua configuração. No bloco digital é feita a configuração do bloco analógico e a contagem de número de pulsos digitais provenientes do bloco analógico.

Ele pode ser configurado como: um comparador entre duas entradas analógicas, um comparador entre uma entrada analógica e uma tensão de referência interna, ou como sensor capacitivo usando um oscilador de relaxação formado no bloco analógico.

Nas Figuras 15 e 16 são mostrados o bloco digital e o bloco analógico do sensor capacitivo. A saída ff_cmp exibida nas Figuras 15 e 16 pode ser acessada através da saída digital SD8 através de configuração dos registradores do bloco de Entradas e Saídas Discretas (ver 8.3).

Figura 15: Bloco digital Sensor Capacitivo

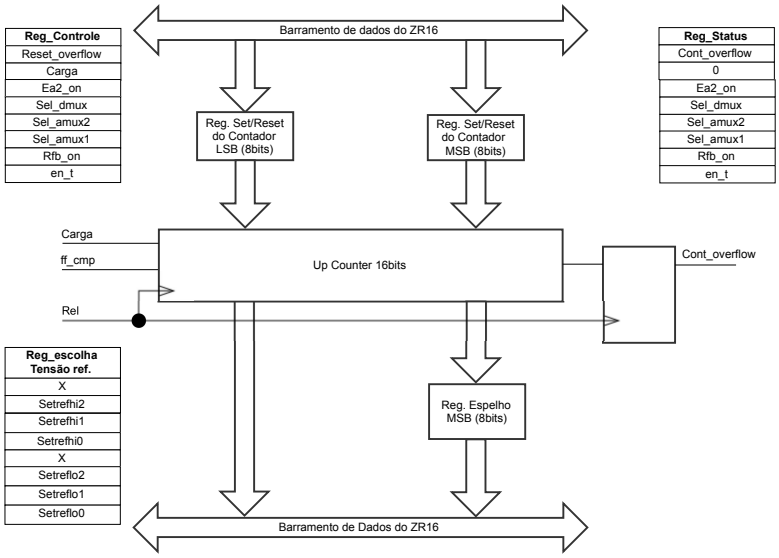
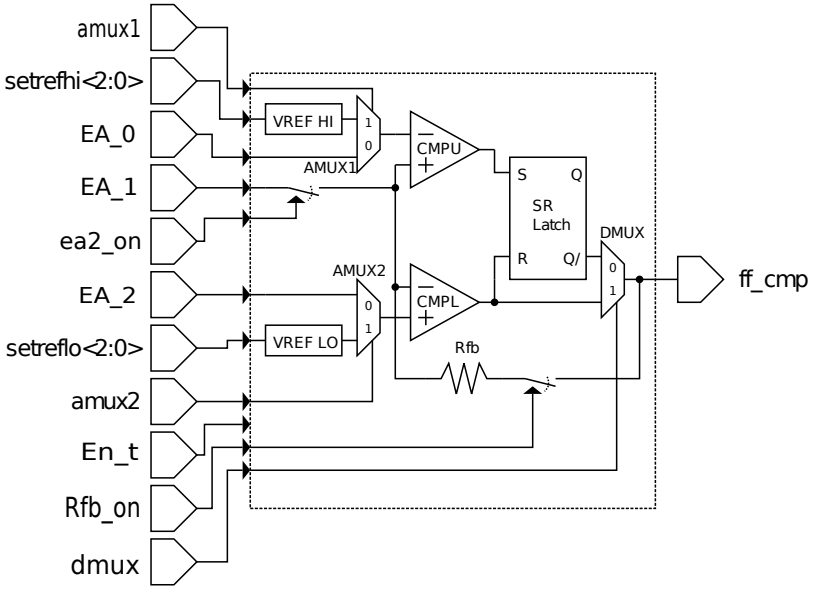


Figura 16: Bloco analógico do Sensor Capacitivo



Através do Registrador de Controle pode-se fazer as seguintes programações no bloco apresentado na Figura 15:

- A conexão/desconexão da entrada analógica EA2 aos comparadores (CMPH entrada não inversora e CMPL entrada inversora)
- Ligar ou desligar a resistência de realimentação (Rfb)
- A seleção do sinal analógico para a entrada inversora do comparador CMPH (VREF HI ou EA1)
- A seleção do sinal analógico para a entrada não inversora do comparador CMPL (VREF LO ou EA3)
- A seleção da fonte do sinal ff_cmp do sensor (o sinal /Q do flip-flop ou o a saída do comparador CMPL)

O *sensor capacitivo* possui um *Up Counter* de 16 bits. Este contador é incrementado toda vez que a entrada ff_cmp trocar o nível lógico de baixo para alto.

O *reset* do *Up Counter* é o sinal n_reset_ucounter, onde en_t é um bit do Registrador de Controle. O bloco possui um registro do overflow do contador. O *reset* deste registro é n_reset_overflow = 0, onde reset_overflow e en_t são bits do Registrador de Controle. O bit de overflow uma vez ligado só é desligado por *reset* do bloco,por uma escrita, em IO no Registrador de Controle, com o reset_overflow = 1 ou com en_t = 0.

Registradores utilizados:

- Registrador de Escolha da Tensão de Referência: **0x12** (leitura/escrita)
- Registrador *Up Counter* - LSBs: **0x13** (leitura)
- Registrador Espelho do Contador - MSBs: **0x14** (leitura)
- Registrador de Controle: **0x15** (escrita)
- Registrador de Status: **0x15** (leitura)
- Registrador de Set/Reset do Contador - LSBs: **0x16** (escrita)
- Registrador de Set/Reset do Contador - MSBs: **0x17** (escrita)

Registrador de Escolha da Tensão de Referência

	Endereço: 0x12							
	Tipo: leitura/escrita							
Índice	7	6	5	4	3	2	1	0
Bit	x	setrefhi2	setrefhi1	setrefhi0	x	setreflo2	setreflo1	setreflo0
Reset	0	0	0	0	0	0	0	0

No Registrador de Escolha da Tensão de Referência pode-se programar os níveis de tensão de referência VREF LO e VREF HI para os comparadores CMPL e CMPH. Onde:

- setrefhi2 a setrefhi0: níveis de comparação VREF HI para o comparador CMPU no bloco;
- setreflo0 a setreflo0: níveis de comparação VREF LO para o comparador CMPL no bloco;
- x: *don't care*.

Tabela 9: Niveis de tensão Vref-LO e Vref-HI de acordo com registrador 0x12.

Vref-LO	**Vref-HI**	**Voltage(V)**
000	000	0.000
001	001	0.471
010	010	0.942
011	011	1.410
100	100	1.880
101	101	2.370
110	110	2.820
111	111	3.300

Registrador *Up Counter* - LSBs

	Endereço: 0x13							
	Tipo: leitura							
Índice	7	6	5	4	3	2	1	0
Bit	inf7	inf6	inf5	inf4	inf3	inf2	inf1	inf0
Reset	0	0	0	0	0	0	0	0

onde:

- inf[7:0]: Contém o byte inferior (bits 7 a 0) do *Up Counter*.

A leitura no endereço de IO X"13"força a transferência dos bits mais significativos (MSBs) do *Up Counter* e do seu bit de *overflow*, respectivamente, para o Registrador Espelho - MSBs e para bit cont_overflow do Registrador de *Status*.

Registrador Espelho do Contador - MSBs

	Endereço: 0x14							
	Tipo: leitura							
Índice	7	6	5	4	3	2	1	0
Bit	sup15	sup14	sup13	sup12	sup11	sup10	sup9	sup8
Reset	0	0	0	0	0	0	0	0

onde:

- sup[15:8]: Contendo o byte superior (bits 15 a 8) do contador.

Registrador de Controle

	Endereço: 0x15							
	Tipo: escrita							
Índice	7	6	5	4	3	2	1	0
Bit	rst_overflow	carga	ea2_on	dmux	amux2	amux1	rfb_on	en_t
Reset	0	0	0	0	0	0	0	0

Onde:

- rst_overflow (este bit não é armazenado):
  - 1: apaga o bit de *overflow* do contador;

  - 0: mantém o bit de *overflow* do contador.
- carga (este bit não é armazenado):
  - 1: carrega o conteúdo do Registrador de Set/Reset do Contador - LSBs e do Registrador de Set/Reset do Contador - MSBs, respectivamente nos bytes inferior e superior do contador, para que a carga seja efetuada é necessário também que en_t =1;
  - 0: mantém o valor atual do contador.
- ea2_on
  - 0: Desconecta a entrada EA2 aos comparadores analógicos
  - 1: Conecta a entrada EA2 dos comparadores analógicos
- dmux: Seleção da saída digital ff_cmp do mux DMUX no bloco.
  - 0: ff_cmp = /Q do *Latch*
  - 1: ff_cmp = saída do do comparador CMPL
- amux2: Seleção da saída analógica do mux AMUX2 no bloco:
  - 0: entrada analógica entrada EA3 do ZR16
  - 1: tensão de referência VREF-LO
- amux1: Seleção da saída analógica do mux AMUX1 no bloco:
  - 0: entrada analógica entrada EA2 do ZR16
  - 1: tensão de referência VREF HI
- rfb_on: conecta/desconeta o resistor de realimentação Rfb no bloco:
  - 1: conecta o resistor de realimentação rfb
  - 0: desconecta o resistor de realimentação rfb
- en_t: Habilitador do contador, *reset* do *Up Counter* e do overflow :
  - 1: contador habilitado , *Up Counter* e bit de *overflow* operando;
  - 0: contador desabilitado , *Up Counter* e bit de *overflow* em *reset*.
- x: *don't care*

Registrador de Status

	Endereço: 0x15							
	Tipo: leitura							
Índice	7	6	5	4	3	2	1	0
Bit	overflow	carga	ea2_on	dmux	amux2	amux1	rfb_on	en_t
Reset	0	0	0	0	0	0	0	0

Onde:

- overflow:bit de overflow do contador;
- ea2_on: bit de seleção de conexão ou desconexão da EA2;
- dmux: bit de seleção da saída digital ff_cmp no mux DMUX do bloco;
- amux2: bit de seleção da saída analógica do mux AMUX2 do bloco;
- amux1: bit de seleção da saída analógica do mux AMUX1 do bloco;
- rfb_on: bit de controle da chave que liga o resistor de realimentação Rfb do bloco;
- en_t: bit de de habilitação/desabilitação do sensor, *reset* do *Up Counter* e do bit de *overflow*;

Registrador de Set/Reset do Contador - LSBs

	Endereço: 0x16							
	Tipo: escrita							
Índice	7	6	5	4	3	2	1	0
Bit	sr_7	sr_6	sr_5	sr_4	sr_3	sr_2	sr_1	sr_0
Reset	0	0	0	0	0	0	0	0

Onde:

- Para i = 0 a 7.
- sr_i = 1: ligar o set do bit equivalente do byte inferior do contador se carga = 1.
- sr_i = 0: ligar o reset do bit equivalente do byte inferior do contador se carga = 1.

Registrador de Set/Reset do Contador - MSBs

	Endereço: 0x17							
	Tipo: escrita							
Índice	7	6	5	4	3	2	1	0
Bit	sr_15	sr_14	sr_13	sr_12	sr_11	sr_10	sr_9	sr_8
Reset	0	0	0	0	0	0	0	0

Onde:

- Para i = 8 a 15.
- sr_i =1: ligar o set do bit equivalente do byte superior do contador se carga = 1.
- sr_i =0: ligar o reset do bit equivalente do byte superior do contador se carga = 1.

8.5 *Watchdog*

O *Watchdog* é utilizado na recuperação do sistema de uma falha do hardware ou de software. Estando o *Watchdog* habilitado, o seu registrador de controle deve ser reinicializado antes do tempo programado. Se isto não ocorrer o microcontrolador será reinicado.

O periférico *Watchdog* tem um contador que é incrementado a cada ciclo do relógio, enquanto o bit hab_reset do Registrador de Controle de Tempo do *Watchdog* estiver ativo (nível lógico alto). Quando o microprocessador executa uma escrita no Registrador de Controle de Tempo, via barramento de IO, o contador juntamente com a *flag overflow* do Registrador de Status são zerados. Essa escrita deve ocorrer dentro de um período determinado (20ms, 40ms, 80ms ou 160ms), caso contrário será ligado o *flag* de *overflow* do *Watchdog* e consequentemente o microcontrolador será reinicializado.

Durante um *reset* gerado pelo *Watchdog* o único registrador que mantém o seu valor é o Registrador de Status deste bloco. Com isso, é possível consultar a *flag overflow* e verificar se o reset foi gerado pelo *Watchdog*.

Os registradores utilizados são:

- Registrador de Controle: **0x1A** (escrita)
- Registrador de Status: **0x1A** (leitura)

Registrador de Controle

	Endereço: 0x1A						
	Tipo: escrita						
Índice	7	6	5	4	3	2	1	0
Bit	hab_reset	x					tmp1	tmp0
Reset	0	x					0	0

Onde:

- hab_reset:
  - 1: habilita o *Watchdog*
  - 0: desabilita o *Watchdog*
- tmp1, tmp0:
  - 0,0: tempo de 20 ms para *overflow*
  - 0,1: tempo de 40 ms para *overflow*
  - 1,0: tempo de 80 ms para *overflow*
  - 1,1: tempo de 160 ms para *overflow*

Registrador de Status

	Endereço: 0x1A							
	Tipo: leitura							
Índice	7	6	5	4	3	2	1	0
Bit	overflow	0						
Reset	0	0						

Onde:

- *overflow*:
  - 1: tempo para escrita no Registrador de Controle de Tempo foi excedido (gerou *reset* no microcontrolador pelo *Watchdog*).
  - 0: tempo para escrita no Registrador de Controle de Tempo não foi excedido.

8.6 *Timer*

O *Timer* disponibiliza a função de temporização para o ZR16S08. Ele pode ser programado para prover contagem de tempo entre 1μs e $\approx$ 0,0655s (1μs x 2^{16}-1). Nele é configurável a recarga automática do *Timer* e a geração de interrupção depois de decorrido o tempo programado. O núcleo do contador possui um *Down Counter* de 16 bits que é decrementado a cada 1μs sempre que a *flag h_timer* estiver habilitada no Registrador de Controle.

Para programar o tempo que se deseja é necessário escrever no Registrador de Carga 0 e no Registrador de Carga 1. Quando o Microprocessador escrever no Registrador de Carga 1 o *Down Counter* é carregado com o conteúdo do Registrador de Carga 0 e do Registrador de Carga 1. Uma vez habilitado o *Down Counter* (*h_timer* = 1 do Registrador de Controle), ele decrementa em 1 o seu conteúdo a cada 1μs. Quando *Down Counter* atinge o zero, isto é memorizado (*pz_mem* do Registrador de Status). Se o Timer estiver no modo de recarga automática (bit ra = 1 do Registrador de Controle) o conteúdo do Registrador de Carga 0 e do Registrador de Carga 1 é carregado no *Down Counter* e este continuará contando. Caso contrário, o bit h_timer é desativado no Registrador de Controle, e consequentemente, encerrando a contagem.

O Microprocessador pode ler contagens parciais do *Down Counter*. Para isto, o Microprocessador lê o Registrador *Down Counter* LSBs, que contém o byte menos significavo do *Down Counter*. Esta leitura força a carga do Registrador Espelho com o byte mais significativo do *Down Counter*, que pode ser lido pelo Microprocessador no próximo acesso. Quando o bit *h_int* do Registrador de Controle estiver ativo e *Down Counter* atingir o zero, uma interrupção será gerada no Microprocessador. Para maiores informações sobre interrupções veja 8.1. O bit *pz_mem* pode ser desativado com uma escrita no Registrador de Controle com o bit desl_pz em 1. No Registrador de *Status* pode ser lido a memorização da passagem por zero (pz_mem) e o estado dos bits de controle: ra (recarga automática), *h_int* (habilitação de interrupção) e *h_timer* (habilitação do timer).

Registradores utilizados:

- Registrador de Carga 0: **0x1C** (escrita)
- Registrador de Carga 1: **0x1D** (escrita)
- Registrador *Down Counter* - LSBs: **0x1C** (leitura)
- Registrador Espelho: **0x1D** (leitura)
- Registrador de Controle: **0x1E** (escrita)
- Registrador de Status: **0x1E** (leitura)

Registrador de Carga 0

	Endereço: 0x1C							
	Tipo: escrita							
Índice	7	6	5	4	3	2	1	0
Bit	cg7	cg6	cg5	cg4	cg3	cg2	cg1	cg0
Reset	0	0	0	0	0	0	0	0

Onde:

- cg(7-0): 8 bits menos significativos que serão carregados no *Down Counter*

Registrador de Carga 1

	Endereço: 0x1D							
	Tipo: escrita							
Índice	7	6	5	4	3	2	1	0
Bit	cg15	cg14	cg13	cg12	cg11	cg10	cg9	cg8
Reset	0	0	0	0	0	0	0	0

Onde:

- cg(15-8): 8 bits mais significativos que serão carregados no *Down Counter*
- Uma escrita neste registrador provoca a carga no *Down Counter* com o conteúdo do Registrador de Carga 0 e deste registrador (Registrador de Carga 1)

Registrador *Down Counter* - LSBs

	Endereço: 0x1C							
	Tipo: leitura							
Índice	7	6	5	4	3	2	1	0
Bit	dc7	dc6	dc5	dc4	dc3	dc2	dc1	dc0
Reset	0	0	0	0	0	0	0	0

Onde:

- dc(7-0): 8 bits menos significativos do *Down Counter*
- Uma leitura neste registrador provoca a carga dos 8 bits mais significativos do *Down Counter* no Registrador Espelho

Registrador Espelho

	Endereço: 0x1D							
	Tipo: leitura							
Índice	7	6	5	4	3	2	1	0
Bit	dc15	dc14	dc13	dc12	dc11	dc10	dc9	dc8
Reset	0	0	0	0	0	0	0	0

Onde:

- dc(15-8): 8 bits mais significativos copiados do *Down Counter* no momento da leitura no Registrador *Down Counter*.

Registrador de Controle

	Endereço: 0x1E							
	Tipo: escrita							
Índice	7	6	5	4	3	2	1	0
Bit	desl_pz	x				h_timer	h_int	ra
Reset	0	x				0	0	0

Onde:

- *desl_pz* (Este bit não é armazenado):
  - 1: desliga o bit pz_mem
  - 0: mantém o bit pz_mem inalterado
- *h_timer*:
  - 1: habilita o *Down Counter* a decrementar
  - 0: desabilita o *Down Counter* a decrementar
- *h_int*:
  - 1: habilita o *Timer* a interromper o microprocessador
  - 0: desabilita o *Timer* a interromper o microprocessador
- ra:
  - 1: habilita a recarga automática a cada passagem por zero do *Down Counter*
  - 0: recarga automática desabilitada
- x: *don't care*

Registrador de Status

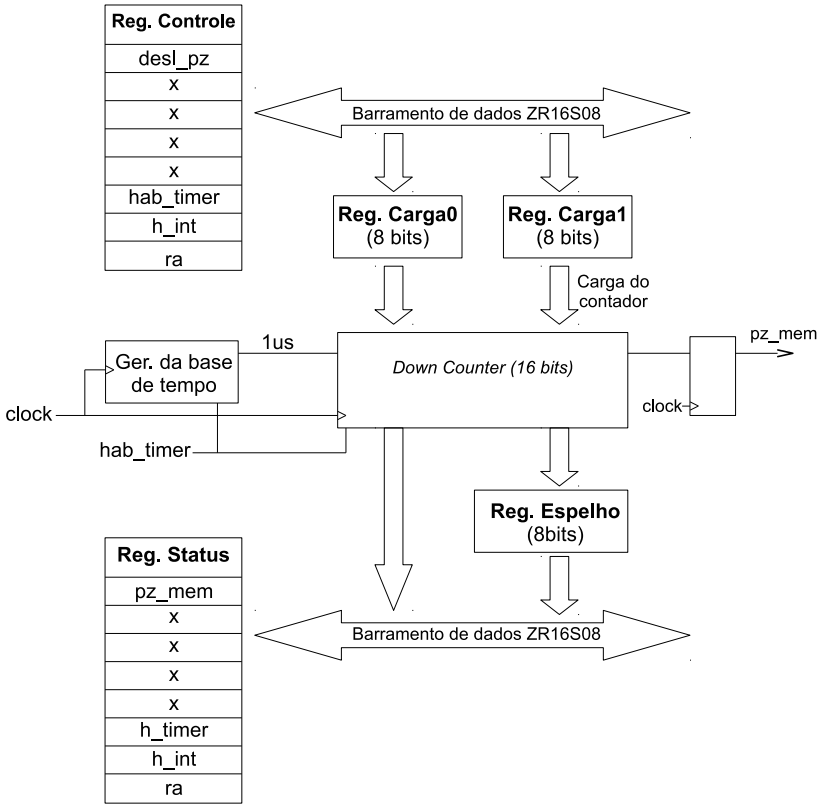
	Endereço: 0x1E							
	Tipo: leitura							
Índice	7	6	5	4	3	2	1	0
Bit	pz_mem	x				h_timer	h_int	ra
Reset	0	x				0	0	0

Onde:

- *pz_mem*: memória da passagem por zero do *Down Counter*
- *h_timer*: informa o *status* do bit h_timer do Registrador de Controle
- *h_int*: informa o *status* do bit h_int do Registrador de Controle
- ra: informa o *status* do bit ra do Registrador de Controle
- x: *don't care*

Diagrama de blocos de funcionamento do TIMER:

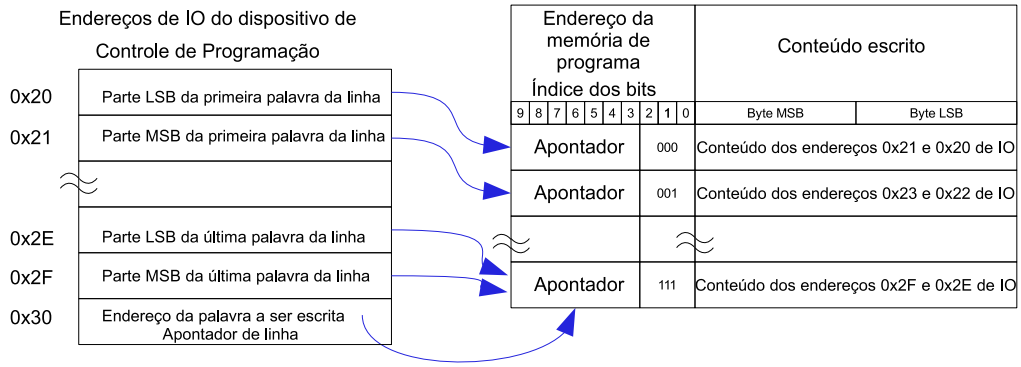
Figura 17: Diagrama de blocos do TIMER



8.7 Controle de Programação

Usando o periférico de controle de programação do ZR16S08 pode-se escrever qualquer parte da memória de programa durante a operação, esta funcionalidade permite que dados sejam armazenados de forma não volátil. O conteúdo de toda uma linha deve ser gravado por vez (uma linha são 8 palavras da memória de programa, cada palavra tem 2 bytes). O ZR16S08 tem disponível 16 registradores de 8 bits onde deve ser armazenado o conteúdo de uma linha. Para realizar essa operação se faz escritas nos endereços de I0 de 0x20 até 0x2F. O efetivo disparo desta operação ocorre com a escrita no Registrador Apontador da linha (endereço 0x30). A escrita na EEPROM tem a duração de 40,5 ms, após esta operação o sistema é resetado e inicializa a execução na posição 0x000 da memória de programa.

Figura 18: Gravação de palavras na memória de programa a partir do dispositivo de Controle de Programação



Byte Inferior Palavra 0

	Endereço: 0x20							
	Tipo: escrita							
Índice	7	6	5	4	3	2	1	0
Bit	cg7	cg6	cg5	cg4	cg3	cg2	cg1	cg0
Reset	0	0	0	0	0	0	0	0

onde:

- $cg_{7,0}$: 8 bits menos significativos que serão carregados no endereço"000"da linha da EEPROM indicada no Registrador X"30".

Byte Superior Palavra 0

	Endereço: 0x21							
	Tipo: escrita							
Índice	7	6	5	4	3	2	1	0
Bit	cg15	cg14	cg13	cg12	cg11	cg10	cg9	cg8
Reset	0	0	0	0	0	0	0	0

onde:

- $cg_{15,8}$: 8 bits mais significativos que serão carregados no endereço bit"000"da linha da EEPROM indicada no Registrador X"30".

Da mesma forma:

- a palavra 1 é carregada nos registradores 0x22 e 0x23;
- a palavra 2 é carregada nos registradores 0x24 e 0x25;
- a palavra 3 é carregada nos registradores 0x26 e 0x27;
- a palavra 4 é carregada nos registradores 0x28 e 0x29;
- a palavra 5 é carregada nos registradores 0x2A e 0x2B;
- a palavra 6 é carregada nos registradores 0x2C e 0x2D;
- a palavra 7 é carregada nos registradores 0x2E e 0x2F.

Registrador Apontador da linha

	Endereço: 0x30							
	Tipo: escrita							
Índice	7	6	5	4	3	2	1	0
Bit	x	end6	end5	end4	end3	end2	end1	end0
Reset	0	0	0	0	0	0	0	0

onde:

- $end_{6,0}$: 7 bits de endereço da linha da EEPROM para a gravação dos dados presentes nos Registradores 0x20 a 0x2F.

A escrita no registrador de endereço 0x30 interrompe a execução do programa principal e realiza escrita dos registradores 0x20 a 0x2F, na EEPROM. Após a escrita na EEPROM ocorre um *reset* do sistema iniciando o programa novamente da posição 0x000.

9 Especificação elétrica

9.1 Valores máximos absolutos:

Tensão de operação V_{DD}	5,0V a 28V
Tensão para programação V_{mp}	6,5V a 7,5V
Tensão de entrada I/Os	V_{GND}-0,5V a V_{REG} + 0,5V
Tensão de saída I/Os	V_{GND}-0,5V a V_{REG} + 0,5V
Temperatura de operação da junção [1]	-40^{o}C a 125^{o}C
Temperatura de armazenagem	-65^{o}C a 150^{o}C
Descarga eletrostática (HBM)[2]	1kV
Dissipação total de potência[3]	0,36W
Corrente máxima de saída por cada I/Os[4]	2mA

[1] Com consumo de corrente nominal.
[2] Usando o modelo *Human Body Model* (HBM).
[3] A temperatura ambiente de 25^{o}C.
Calculada como: $I_{DC} * V_{DDmax} + I_{OUT} * (V_{DDmax} - V_{REG}) + 3 \cdot I_{SOD} * V_{SOD}$
para $I_{OUT} = 0$
[4] Limitada à corrente fornecida pelo regulador $I_{OUT} - I_{DC}$.
Não aplica para os pinos de saída DAC0, DAC1, DAC2.
Não aplica para os pinos EA0/SD7, EA1/SD6, EA2/SD5, EA3/SD4.

9.2 Caracteristícas DC

Símbolo	**Parâmetro**	**Condições do teste**	**Mín.**	**Típ.**	**Máx.**	**Uni.**
V_{DC}	Fonte de alimentação analógica[1]		5		28	V
I_{DC}	Consumo de corrente típico [2]	$V_{DC} = 5V$ $V_{DC} = 28V$		1,79 1,97		mA mA
V_{il}	Tensão de entrada baixa	V_{REG}=3.3V[(3)]	–	–	0,99	V
V_{ih}	Tensão de entrada alta	V_{REG}=3.3V[(3)]	2,31	–	–	V
V_{ol}	Tensão de saída baixa	V_{REG}=3.3V[(3)]	–	–	0,66	V
V_{oh}	Tensão de saída alta	V_{REG}=3.3V[(3)]	2,64	–	–	V
I_{ol}	Corrente de saída baixa	20% V_{REG} [5] 10% V_{REG} [4]	–	–	2 1.95[(4)]	mA mA
I_{oh}	Corrente de saída alta	80% V_{REG} [5] 90% V_{REG} [4]	–	–	2 1.44[(4)]	mA mA

[1] Parâmetro equivalente a V_{in} (Ver Tabela 9.3).
[2] Consumo típico médio. Altamente dependente do programa executado e dos periféricos ativos.
[3] Parâmetro definido na Tabela 9.3.
[4] Parâmetros I_{ol}, I_{oh} que se aplicam para os pinos EA0/SD7, EA1/SD6, EA2/SD5, EA3/SD4.
[5] Parâmetro definido para as demais saídas.

9.3 Regulador de tensão

Símbolo	Parâmetro	Condições do teste	Mín.	Típ.	Máx.	Uni.
V_{in}	Fonte de alimentação analógica		5		28	V
V_{REG}	Tensão de regulação	V_{in}=5V	3,21	3,28	3,38	V
I_{OUT}	Corrente de saída				5-I_{DC}	mA
REG_{carga}	Regulação de carga	I_{out}=0 a 5mA	0,015			%/mA
REG_{linha}	Regulação de linha	V_{in}=5V a 6V	0,2			mV/V
$PSRR_{REG}$	Fator de rejeição à fonte de alimentação	Frequência= 100kHz		-76		dB
C_O	Capacitor na saída		1	10	–	μF
R_{esr}	Resistência ESR (1)		–	–	100	$m\Omega$
A menos que o contrário seja estabelecido: T_{amb}=25^oC e V_{REG}=3,3V. (1) Resistencia ESR: Resistência equivalente em serie do capacitor conectado na saída.						

9.4 Gerador interno de relógio

Símbolo	Parâmetro	Condições do teste	Mín.	Típ.	Máx.	Uni.
f_{osc}	Frequência de oscilação	Ajustada internamente [1]	11,4	12,0	12,6	MHz
RZ	Razão cíclica *(Duty cycle)*			54		%
A menos que o contrário seja estabelecido: T_{amb}=25^oC e V_{REG}=3,3V [1] $f_{clk} = f_{osc}/3$, sendo f_{clk} a frequência de operação do sistema						

9.5 Power on Reset

Símbolo	Parâmetro	Condições do teste	Mín.	Típ.	Máx.	Unit
V_{POR}	Tensão limiar do POR			3		V
$V_{POR_{hys}}$	Histerese do POR			100		mV
A menos que o contrário seja estabelecido: T_{amb}=25^oC e V_{REG}=3.3V						

9.6 Conversor ADC

Símbolo	Parâmetro	Condições do teste	Mín.	Típ.	Máx.	Unit
V_{ADCI}	Tensão do entrada do ADC		0		V_{REG}	V
INL	*Integral Non-linearity*		-5	0	5	LSB
DNL	*Differential Non-linearity*		-0,7	0	0,2	LSB
V_{ADCOS}	Offset do conversor ADC		-8	1	24	mV
f_{ADC_s}	Taxa de conversão			40,0		ksps(1)
I_{ADCD}	Consumo de corrente adicional se o ADC é habilitado			105		μA
R_{IN}	Resistência de entrada			13		kΩ
C_{IN}	Capacitância de entrada			8		pF
A menos que o contrário seja estabelecido: T_{amb}=25^oC e V_{REG}=3.3V (1) Mil conversões por segundo						

9.7 Sensor Capacitivo

Símbolo	Parâmetro	Condições do teste	Mín.	Típ.	Máx.	Unit
V_{REF_HI}	Referência superior		$V_{REG}/8$	–	V_{REG}	V
V_{REF_LO}	Referência inferior		$V_{REG}/8$	–	V_{REG}	V
f_{SCAP}	Frequência de oscilação do sensor capacitivo	Sem qualquer capacitância ligada ao pino EA1 SD6		446		kHz
R_{fb}	Resistência de realimentação interna		–	45	–	kΩ
A menos que o contrário seja estabelecido: T_{amb}=25^oC e V_{REG}=3,3V						

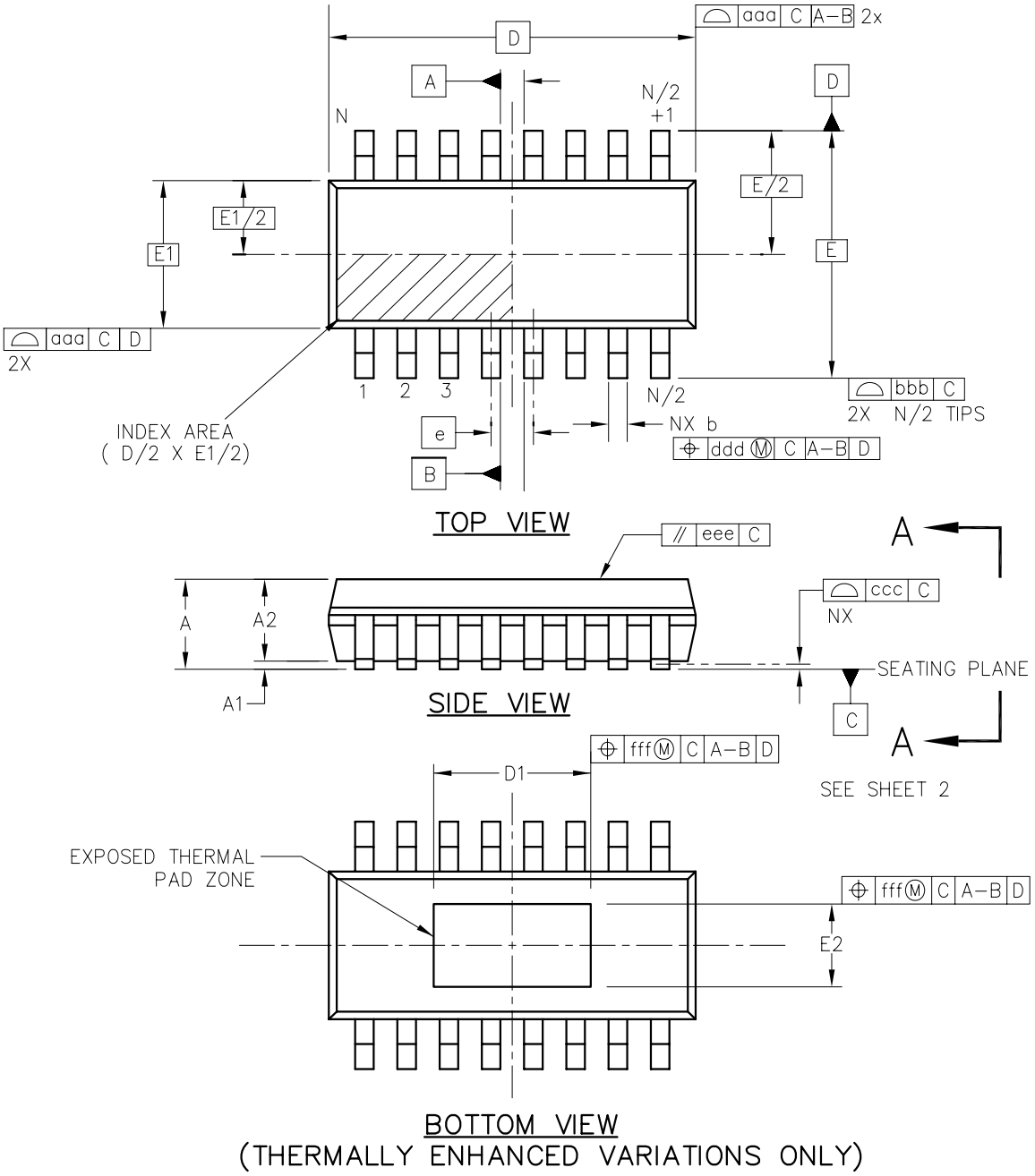
9.8 DAC

Símbolo	Parâmetro	Condições do teste	Mín.	Típ.	Máx.	Unid
I_{DAC_l}	Corrente de fuga	Medido @ $V_{DAC} = 5V$		43		μA
V_{DAC}	Tensão nos pinos de saída				28	V
$I_{DAC_{full}}$	Capacidade de corrente na saída a escala total		19,2	22,7	25,7	mA
$I_{DAC_{half}}$	Capacidade de corrente na saída a meia escala		9,5	11,0	12,7	mA
$I_{DAC_{step}}$	Passo da corrente na saída do driver			1,25		mA
R_{on}	Resistência de ativação	I_{DAC}=29mA	9	13	19	Ω
A menos que o contrário seja estabelecido: T_{amb}=25^oC e V_{REG}=3,3V						

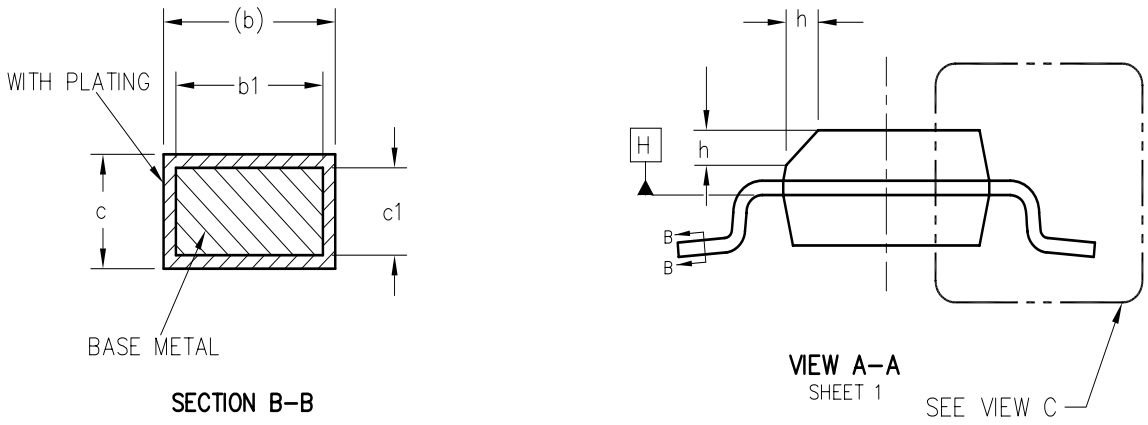
10 Encapsulamento

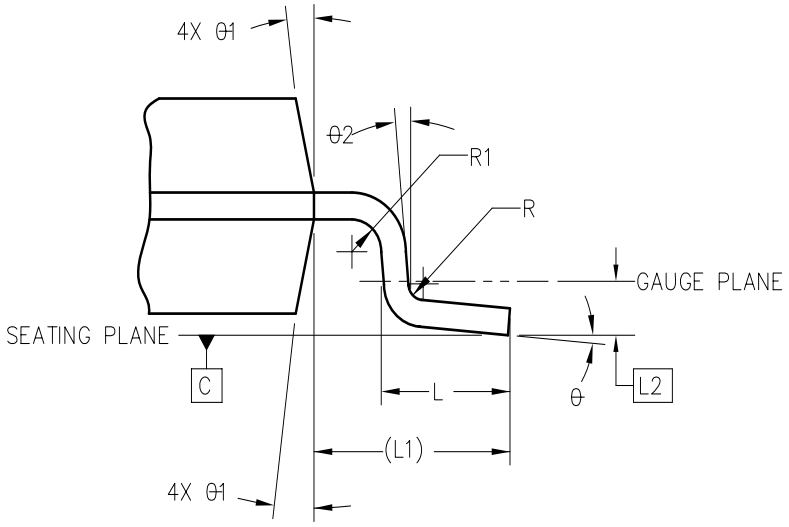
Dimensionamento e tolerâncias ASME Y14.5-1994.

Dimensões em milimetros e ângulos em graus.

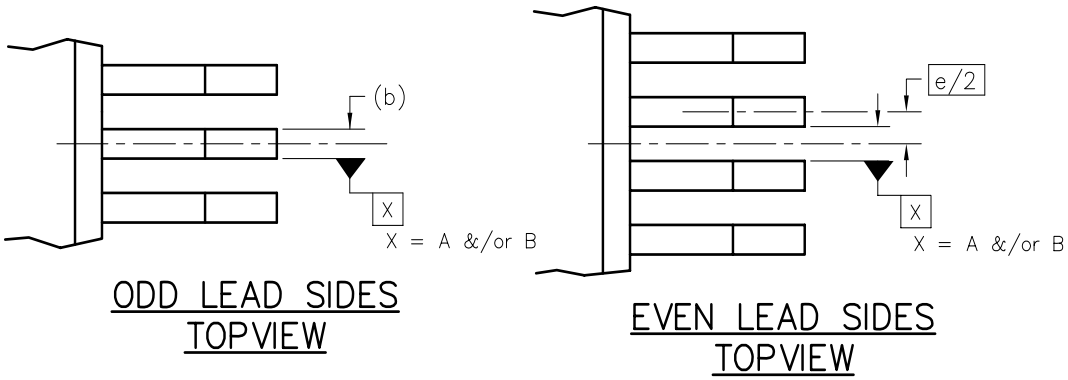


JEDEC SOLID STATE PRODUCT OUTLINE / PLASTIC SMALL OUTLINE FAMILY 1,27mm PITCH, 3,90mm BODY WIDTH / ISSUE F / AUGUST,2008





VIEW C



JEDEC SOLID STATE PRODUCT OUTLINE / PLASTIC SMALL OUTLINE FAMILY 1,27mm PITCH, 3,90mm BODY WIDTH / ISSUE F / AUGUST,2008

SYMBOL	COMMON DIMENSIONS			NOTE
	MIN	NOM	MAX	
A2	1.25	–	–	
b	0.31	–	0.51	7,8
b1	0.28	–	0.48	7,8
c	0.10	–	0.25	7
c1	0.10	–	0.23	7
E	6.00 BSC			
E1	3.90 BSC			3,4
e	1.27 BSC			
L	0.40	–	1.27	
L1	1.04 REF			
L2	0.25 BSC			
R	0.07	–	–	
R1	0.07	–	–	
h	0.25	–	0.50	9
θ	0°	–	8°	
θ1	5°	–	15°	
θ2	0°	–	–	
NOTES	1,2			
REF	11–719S, 11–801S			
ISSUE	F			

SYMBOL	TOLERANCES OF FORM AND POSITION	NOTE
aaa	0.10	
bbb	0.20	
ccc	0.10	
ddd	0.25	
eee	0.10	
fff	0.15	
NOTES	1,2	
REF	11–719S	
ISSUE	E	

VARIATION TABLES

STANDARD

VARIATION ► / SYMBOL ▼		AA	AB	AC	NOTE
A	MIN	---			
	NOM	---			
	MAX	1.75			
A1	MIN	0.10			10
	NOM	---			
	MAX	0.25			
D BSC		4.90	8.65	9.90	3,4
N		8	14	16	6
NOTES		1,2			
REF		11–719S			
ISSUE		E			

THERMAL

VARIATION ► / SYMBOL ▼		BA	BB	BC	NOTE
A	MIN	---			
	NOM	---			
	MAX	1.70			
A1	MIN	0.00			10
	NOM	---			
	MAX	0.15			
D BSC		4.90	8.65	9.90	3,4
D1	MIN	1.50			11
	NOM	---			
	MAX	---			
E2	MIN	1.00			11
	NOM	---			
	MAX	---			
N		8	14	16	6
NOTES		1,2			
REF		11–719S			
ISSUE		E			

JEDEC SOLID STATE PRODUCT OUTLINE / PLASTIC SMALL OUTLINE FAMILY 1,27mm PITCH, 3,90mm BODY WIDTH / ISSUE F / AUGUST,2008

Apêndices

A Apêndice A

A.1 Historico Revisão

Versão Preliminar.	v1.0		18/DEZ/2014
Versão Inicial.	v1.1	Alterado referências da figura do bloco ADC	19/DEZ/2014
Versão	v1.2	Incluido as Listas de Figuras e Tabelas	07/JAN/2015
Versão	v1.3	Correções de textos	29/JAN/2015
Versão	v1.4	Alteração do layout	04/FEV/2015
Versão	v1.5	Aletado figuras sensor capasitivo e instução SHL	19/MAI/2015
Versão	v1.6	Incluido Nota dobre interrupção durante instrução com MP=1	02/JUL/2015

B Apêndice B

B.1 Lista de Figuras

B.2 Lista de Tabelas