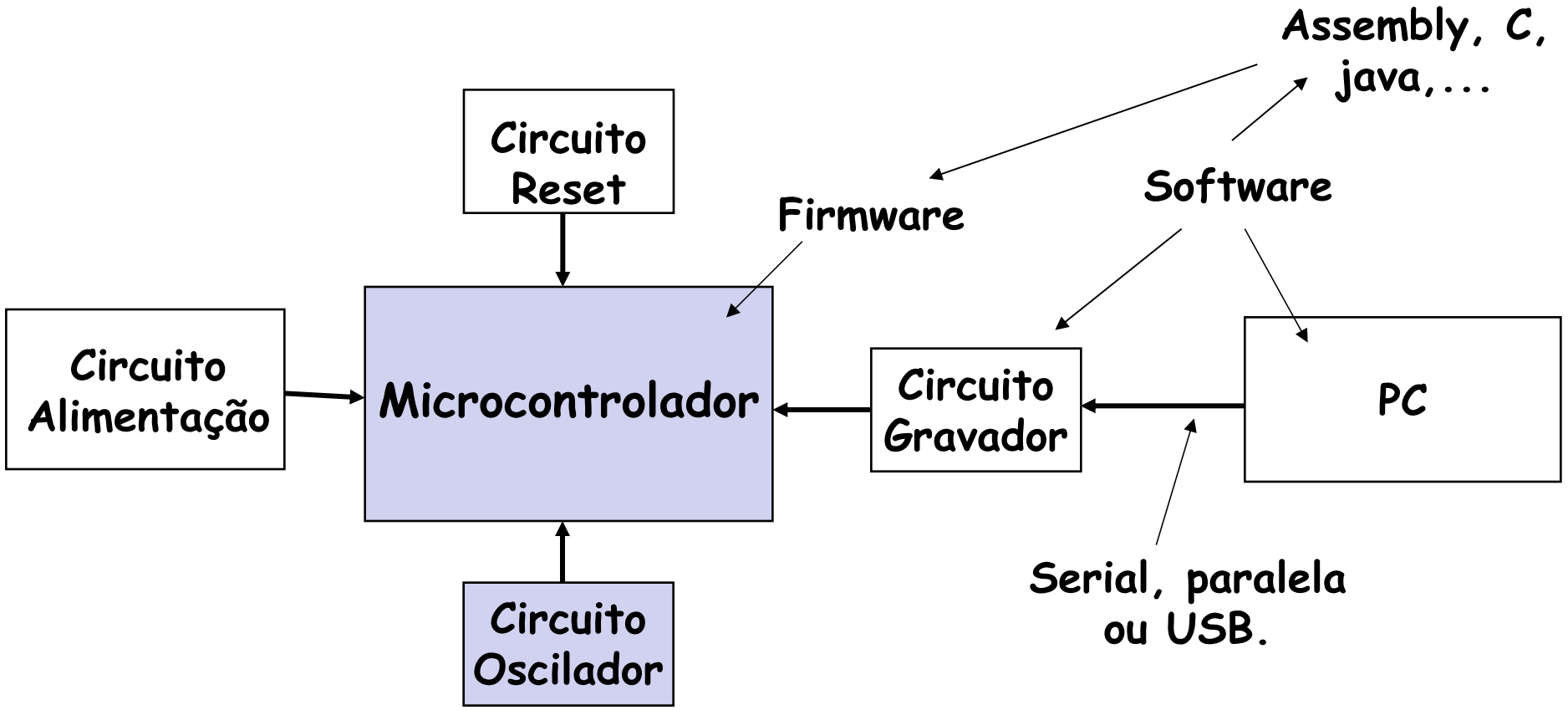
AVR CPU Core

Jadsonlee da Silva Sá

Jadsonlee.sa@univasf.edu.br
www.univasf.edu.br/~jadsonlee.sa

Introdução aos Sistemas Microcontrolados

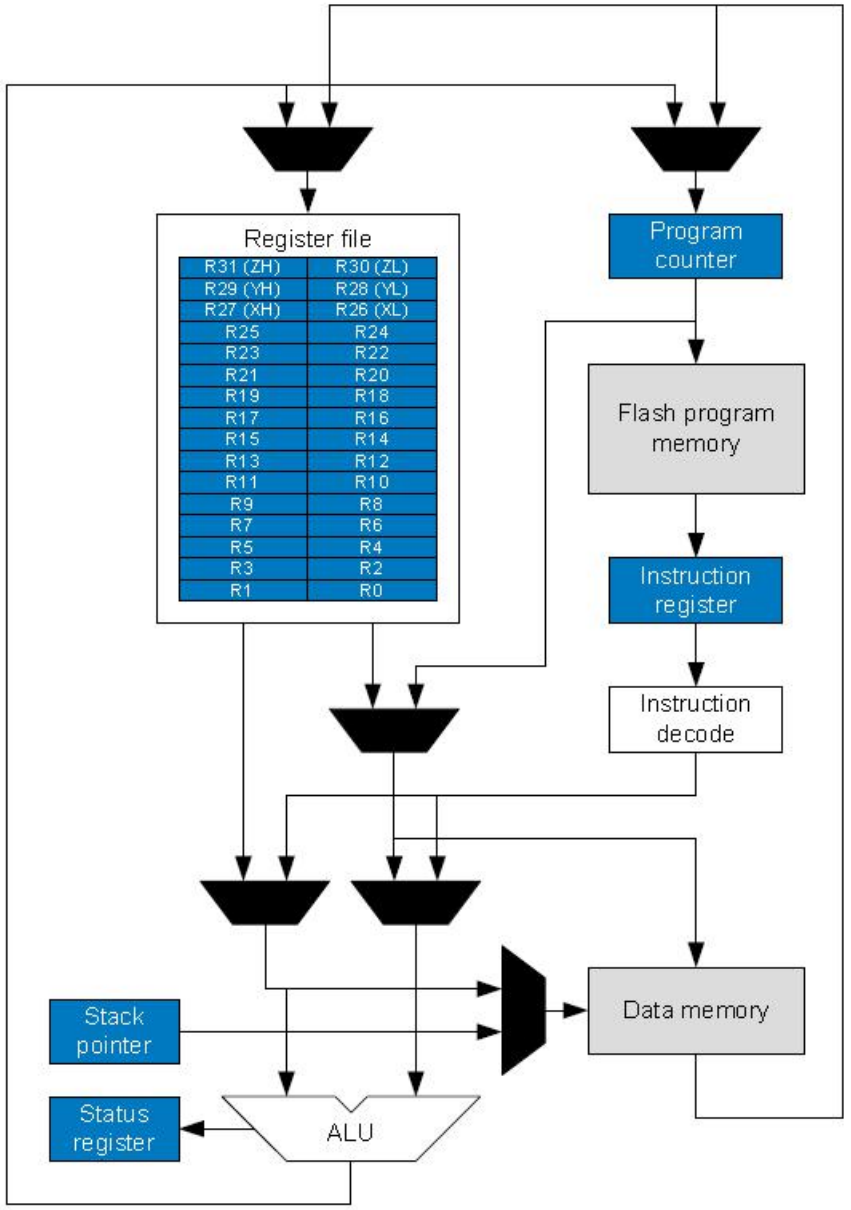


AVR CPU Core

Arquitetura Harvard.

Arquivo de registradores possui 32 registradores de 8 bits de propósito geral.

As operações da ULA são divididas em três principais categorias: aritmética, lógica e bit.



AVR CPU Core - ULA

ARITHMETIC AND LOGIC INSTRUCTIONS					
Mnemonics	Operands	Description	Operation	Flags	#Clocks
ADD	Rd, Rr	Add two Registers without Carry	Rd ← Rd + Rr	Z,C,N,V,H	1
ADC	Rd, Rr	Add two Registers with Carry	Rd ← Rd + Rr + C	Z,C,N,V,H	1
ADIW	Rdl,K	Add Immediate to Word	Rdh:Rdl ← Rdh:Rdl + K	Z,C,N,V,S	2
SUB	Rd, Rr	Subtract two Registers	Rd ← Rd - Rr	Z,C,N,V,H	1
SUBI	Rd, K	Subtract Constant from Register	Rd ← Rd - K	Z,C,N,V,H	1
SBC	Rd, Rr	Subtract two Registers with Carry	Rd ← Rd - Rr - C	Z,C,N,V,H	1
SBCI	Rd, K	Subtract Constant from Reg with Carry.	Rd ← Rd - K - C	Z,C,N,V,H	1
SBIW	Rdl,K	Subtract Immediate from Word	Rdh:Rdl ← Rdh:Rdl - K	Z,C,N,V,S	2
AND	Rd, Rr	Logical AND Registers	Rd ← Rd · Rr	Z,N,V	1
ANDI	Rd, K	Logical AND Register and Constant	Rd ← Rd · K	Z,N,V	1
OR	Rd, Rr	Logical OR Registers	Rd ← Rd v Rr	Z,N,V	1
ORI	Rd, K	Logical OR Register and Constant	Rd ← Rd v K	Z,N,V	1
EOR	Rd, Rr	Exclusive OR Registers	Rd ← Rd ⊕ Rr	Z,N,V	1
COM	Rd	One's Complement	Rd ← 0xFF - Rd	Z,C,N,V	1
NEG	Rd	Two's Complement	Rd ← 0x00 - Rd	Z,C,N,V,H	1
SBR	Rd,K	Set Bit(s) in Register	Rd ← Rd v K	Z,N,V	1
CBR	Rd,K	Clear Bit(s) in Register	Rd ← Rd · (0xFF - K)	Z,N,V	1
INC	Rd	Increment	Rd ← Rd + 1	Z,N,V	1
DEC	Rd	Decrement	Rd ← Rd - 1	Z,N,V	1
TST	Rd	Test for Zero or Minus	Rd ← Rd · Rd	Z,N,V	1
CLR	Rd	Clear Register	Rd ← Rd ⊕ Rd	Z,N,V	1
SER	Rd	Set Register	Rd ← 0xFF	None	1
MUL	Rd, Rr	Multiply Unsigned	R1:R0 ← Rd × Rr	Z,C	2
MULS	Rd, Rr	Multiply Signed	R1:R0 ← Rd × Rr	Z,C	2
MULSU	Rd, Rr	Multiply Signed with Unsigned	R1:R0 ← Rd × Rr	Z,C	2
FMUL	Rd, Rr	Fractional Multiply Unsigned	R1:R0 ← (Rd × Rr) << 1	Z,C	2
FMULS	Rd, Rr	Fractional Multiply Signed	R1:R0 ← (Rd × Rr) << 1	Z,C	2
FMULSU	Rd, Rr	Fractional Multiply Signed with Unsigned	R1:R0 ← (Rd × Rr) << 1	Z,C	2

Rd:	Destination (and source) register in the Register File
Rr:	Source register in the Register File
R:	Result after instruction is executed
K:	Constant data
k:	Constant address
b:	Bit in the Register File or I/O Register (3-bit)
s:	Bit in the Status Register (3-bit)
X,Y,Z:	Indirect Address Register (X=R27:R26, Y=R29:R28 and Z=R31:R30)
A:	I/O location address
q:	Displacement for direct addressing (6-bit)

SREG:	Status Register
C:	Carry Flag
Z:	Zero Flag
N:	Negative Flag
V:	Two's complement overflow indicator
S:	N ⊕ V, For signed tests
H:	Half Carry Flag
T:	Transfer bit used by BLD and BST instructions
I:	Global Interrupt Enable/Disable Flag

BRANCH INSTRUCTIONS					
Mnemonics	Operands	Description	Operation	Flags	#Clocks
RJMP	k	Relative Jump	PC ← PC + k + 1	None	2
IJMP		Indirect Jump to (Z)	PC ← Z	None	2
JMP(1)	k	Direct Jump	PC ← k	None	3
RCALL	k	Relative Subroutine Call	PC ← PC + k + 1	None	3

BIT AND BIT-TEST INSTRUCTIONS					
Mnemonics	Operands	Description	Operation	Flags	#Clocks
SBI	P,b	Set Bit in I/O Register	I/O(P,b) ← 1	None	2
CBI	P,b	Clear Bit in I/O Register	I/O(P,b) ← 0	None	2

DATA TRANSFER INSTRUCTIONS					
Mnemonics	Operands	Description	Operation	Flags	#Clocks
MOV	Rd, Rr	Move Between Registers	Rd ← Rr	None	1
MOVW	Rd, Rr	Copy Register Word	Rd+1:Rd ← Rr+1:Rr	None	1
LDI	Rd, K	Load Immediate	Rd ← K	None	1
LD	Rd, X	Load Indirect	Rd ← (X)	None	2
LD	Rd, X+	Load Indirect and Post-Increment	Rd ← (X), X ← X + 1	None	2
LD	Rd, - X	Load Indirect and Pre-Decrement	X ← X - 1, Rd ← (X)	None	2
LD	Rd, Y	Load Indirect	Rd ← (Y)	None	2
LD	Rd, Y+	Load Indirect and Post-Increment	Rd ← (Y), Y ← Y + 1	None	2

MCU CONTROL INSTRUCTIONS					
Mnemonics	Operands	Description	Operation	Flags	#Clocks
NOP		No Operation	No Operation	None	1
SLEEP		Sleep	(see specific descr. for Sleep function)	None	1
WDR		Watchdog Reset	(see specific descr. for WDR/timer)	None	1
BREAK		Break	For On-chip Debug Only	None	N/A

BRANCH INSTRUCTIONS					
Mnemonics	Operands	Description	Operation	Flags	#Clocks
ICALL		Indirect Call to (Z)	PC ← Z	None	3
CALL(1)	k	Direct Subroutine Call	PC ← k	None	4
RET		Subroutine Return	PC ← STACK	None	4
RETI		Interrupt Return	PC ← STACK	I	4
CPSE	Rd,Rr	Compare, Skip if Equal	if(Rd = Rr) PC ← PC + 2 or 3	None	1/2/3
CP	Rd,Rr	Compare	Rd - Rr	Z, N,V,C,H	1
CPC	Rd,Rr	Compare with Carry	Rd - Rr - C	Z, N,V,C,H	1
CPI	Rd,K	Compare Register with Immediate	Rd - K	Z, N,V,C,H	1
SBRC	Rr, b	Skip if Bit in Register Cleared	if(Rr(b)=0) PC ← PC + 2 or 3	None	1/2/3
SBRS	Rr, b	Skip if Bit in Register is Set	if(Rr(b)=1) PC ← PC + 2 or 3	None	1/2/3
SBIC	A, b	Skip if Bit in I/O Register Cleared	if(I/O(A,b)=1) PC ← PC + 2 or 3	None	1/2/3
SBIS	A, b	Skip if Bit in I/O Register is Set	if(I/O(A,b)=1) PC ← PC + 2 or 3	None	1/2/3
BRBS	s, k	Branch if Status Flag Set	if(SREG(s) = 1) then PC←PC+k + 1	None	1/2
BRBC	s, k	Branch if Status Flag Cleared	if(SREG(s) = 0) then PC←PC+k + 1	None	1/2
BREQ	k	Branch if Equal	if(Z = 1) then PC ← PC + k + 1	None	1/2
BRNE	k	Branch if Not Equal	if(Z = 0) then PC ← PC + k + 1	None	1/2
BRCS	k	Branch if Carry Set	if(C = 1) then PC ← PC + k + 1	None	1/2
BRCC	k	Branch if Carry Cleared	if(C = 0) then PC ← PC + k + 1	None	1/2
BRSH	k	Branch if Same or Higher	if(C = 0) then PC ← PC + k + 1	None	1/2
BRLO	k	Branch if Lower	if(C = 1) then PC ← PC + k + 1	None	1/2
BRMI	k	Branch if Minus	if(N = 1) then PC ← PC + k + 1	None	1/2
BRPL	k	Branch if Plus	if(N = 0) then PC ← PC + k + 1	None	1/2
BRGE	k	Branch if Greater or Equal, Signed	if(N ⊕ V= 0) then PC ← PC + k + 1	None	1/2
BRLT	k	Branch if Less Than Zero, Signed	if(N ⊕ V= 1) then PC ← PC + k + 1	None	1/2
BRHS	k	Branch if Half Carry Flag Set	if(H = 1) then PC ← PC + k + 1	None	1/2
BRHC	k	Branch if Half Carry Flag Cleared	if(H = 0) then PC ← PC + k + 1	None	1/2
BRTS	k	Branch if T Flag Set	if(T = 1) then PC ← PC + k + 1	None	1/2
BRTC	k	Branch if T Flag Cleared	if(T = 0) then PC ← PC + k + 1	None	1/2
BRVS	k	Branch if Overflow Flag is Set	if(V = 1) then PC ← PC + k + 1	None	1/2
BRVC	k	Branch if Overflow Flag is Cleared	if(V = 0) then PC ← PC + k + 1	None	1/2
BRIE	k	Branch if Interrupt Enabled	if(I = 1) then PC ← PC + k + 1	None	1/2
BRID	k	Branch if Interrupt Disabled	if(I = 0) then PC ← PC + k + 1	None	1/2

AVR CPU Core - ULA

BIT AND BIT-TEST INSTRUCTIONS					
Mnemonics	Operands	Description	Operation	Flags	#Clocks
LSL	Rd	Logical Shift Left	Rd(n+1) ← Rd(n), Rd(0) ← 0	Z,C,N,V	1
LSR	Rd	Logical Shift Right	Rd(n) ← Rd(n+1), Rd(7) ← 0	Z,C,N,V	1
ROL	Rd	Rotate Left Through Carry	Rd(0)←C,Rd(n+1)← Rd(n),C¬Rd(7)	Z,C,N,V	1
ROR	Rd	Rotate Right Through Carry	Rd(7)←C,Rd(n)← Rd(n+1),C←Rd(0)	Z,C,N,V	1
ASR	Rd	Arithmetic Shift Right	Rd(n) ← Rd(n+1), n=0...6	Z,C,N,V	1
SWAP	Rd	Swap Nibbles	Rd(3..0)←Rd(7...4),Rd(7...4)¬Rd(3...0)	None	1
BSET	s	Flag Set	SREG(s) ← 1	SREG(s)	1
BCLR	s	Flag Clear	SREG(s) ← 0	SREG(s)	1
BST	Rr, b	Bit Store from Register to T	T ← Rr(b)	T	1
BLD	Rd, b	Bit load from T to Register	Rd(b) ← T	None	1
SEC		Set Carry	C ← 1	C	1
CLC		Clear Carry	C ← 0	C	1
SEN		Set Negative Flag	N ← 1	N	1
CLN		Clear Negative Flag	N ← 0	N	1
SEZ		Set Zero Flag	Z ← 1	Z	1
CLZ		Clear Zero Flag	Z ← 0	Z	1
SEI		Global Interrupt Enable	I ← 1	I	1
CLI		Global Interrupt Disable	I ← 0	I	1
SES		Set Signed Test Flag	S ← 1	S	1
CLS		Clear Signed Test Flag	S ← 0	S	1
SEV		Set Two's Complement Overflow.	V ← 1	V	1
CLV		Clear Two's Complement Overflow	V ← 0	V	1
SET		Set T in SREG	T ← 1	T	1
CLT		Clear T in SREG	T ← 0	T	1
SEH		Set Half Carry Flag in SREG	H ← 1	H	1
CLH		Clear Half Carry Flag in SREG	H ← 0	H	1

AVR CPU Core - ULA

DATA TRANSFER INSTRUCTIONS					
Mnemonics	Operands	Description	Operation	Flags	#Clocks
LD	Rd, - Y	Load Indirect and Pre-Decrement	Y ← Y - 1, Rd ← (Y)	None	2
LDD	Rd,Y+q	Load Indirect with Displacement	Rd ← (Y + q)	None	2
LD	Rd, Z	Load Indirect	Rd ← (Z)	None	2
LD	Rd, Z+	Load Indirect and Post-Increment	Rd ← (Z), Z ← Z+1	None	2
LD	Rd, -Z	Load Indirect and Pre-Decrement	Z ← Z - 1, Rd ← (Z)	None	2
LDD	Rd, Z+q	Load Indirect with Displacement	Rd ← (Z + q)	None	2
LDS	Rd, k	Load Direct from SRAM	Rd ← (k)	None	2
ST	X, Rr	Store Indirect	(X) ← Rr	None	2
ST	X+, Rr	Store Indirect and Post-Increment	(X) ← Rr, X ← X + 1	None	2
ST	- X, Rr	Store Indirect and Pre-Decrement	X ← X - 1, (X) ← Rr	None	2
ST	Y, Rr	Store Indirect	(Y) ← Rr	None	2
ST	Y+, Rr	Store Indirect and Post-Increment	(Y) ← Rr, Y ← Y + 1	None	2
ST	- Y, Rr	Store Indirect and Pre-Decrement	Y ← Y - 1, (Y) ← Rr	None	2
STD	Y+q,Rr	Store Indirect with Displacement	(Y + q) ← Rr	None	2
ST	Z, Rr	Store Indirect	(Z) ← Rr	None	2
ST	Z+, Rr	Store Indirect and Post-Increment	(Z) ← Rr, Z ← Z + 1	None	2
ST	-Z, Rr	Store Indirect and Pre-Decrement	Z ← Z - 1, (Z) ← Rr	None	2
STD	Z+q,Rr	Store Indirect with Displacement	(Z + q) ← Rr	None	2
STS	k, Rr	Store Direct to SRAM	(k) ← Rr	None	2
LPM		Load Program Memory	R0 ← (Z)	None	3
LPM	Rd, Z	Load Program Memory	Rd ← (Z)	None	3
LPM	Rd, Z+	Load Program Memory and Post-Inc	Rd ← (Z), Z ← Z+1	None	3
SPM		Store Program Memory	(Z) ← R1:R0	None	-
IN	Rd, A	In from I/O Location	Rd ← I/O (A)	None	1
OUT	A, Rr	Out to I/O Location	I/O (A) ← Rr	None	1
PUSH	Rr	Push Register on Stack	STACK ← Rr	None	2
POP	Rd	Pop Register from Stack	Rd ← STACK	None	2

AVR CPU Core – Registrador STATUS

- ✓ Contém informações sobre o resultado das instruções aritméticas executadas.
- ✓ **STATUS** é atualizado após qualquer uma das operações da ULA.
- ✓ Não é automaticamente armazenado quando ocorre uma interrupção e restaurado após o retorno da interrupção. Isso deve ser manipulado por software.

AVR CPU Core – Registrador STATUS

Name: SREG
Offset: 0x5F
Reset: 0x00
Property: When addressing as I/O Register: address offset is 0x3F

Bit	7	6	5	4	3	2	1	0
	I	T	H	S	V	N	Z	C
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit 0 – C (Carry flag).

Bit 1 – Z (Zero flag).

Bit 2 – N (negative flag).

Bit 3 – V (flag de overflow comp. de 2).

Bit 4 – S (Sign flag) – XOR entre N e V.

AVR CPU Core – Registrador STATUS

Name: SREG
Offset: 0x5F
Reset: 0x00
Property: When addressing as I/O Register: address offset is 0x3F

Bit	7	6	5	4	3	2	1	0
	I	T	H	S	V	N	Z	C
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit 5 – H (Half carry).

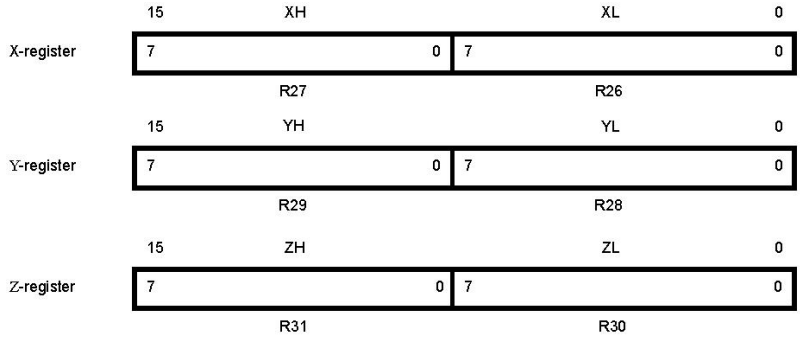
Bit 6 – T (Copy storage) – As instruções BLD (Bit LoaD) e BST (Bit Store) usam o bit T como fonte ou destino para uma operação de bit.

Bit 7 – I (Global interrupt enable) – Se I=1, a interrupção global é habilitada. Se I=0, a interrupção global é desabilitada.

AVR CPU Core – Arquivo de Registrador

- ✓ Registradores de propósito geral.

	7 0	Addr.
General Purpose Working Registers	R0	0x00
	R1	0x01
	R2	0x02
	…	
	R13	0x0D
	R14	0x0E
	R15	0x0F
	R16	0x10
	R17	0x11
	…	
	R26	0x1A
	R27	0x1B
	R28	0x1C
	R29	0x1D
	R30	0x1E
	R31	0x1F



X-register High Byte

Y-register Low Byte

Y-register High Byte

Z-register Low Byte

Z-register High Byte

AVR CPU Core – Pilha

- ✓ Alocada na SRAM de dados geral. O tamanho é limitado pelo tamanho total da SRAM e o uso da SRAM.
- ✓ A pilha é implementada de modo crescente a partir das locações de memória superiores para inferiores.

AVR CPU Core – Pilha

- ✓ O apontador de pilha (SP) é implementado em dois registradores de 8 bits no espaço de E/S.

Name: SPH
Offset: 0x5E
Reset: RAMEND
Property: When addressing I/O Registers as data space the offset address is 0x3E

Bit	7	6	5	4	3	2	1	0
						(SP[10:8]) SPH		
Access						RW	RW	RW
Reset						0	0	0

Bits 2:0 – (SP[10:8]) SPH: Stack Pointer Register
SPH and SPL are combined into SP. It means SPH[2:0] is SP[10:8].

Name: SPL
Offset: 0x5D
Reset: 0x11111111
Property: When addressing I/O Registers as data space the offset address is 0x3D

Bit	7	6	5	4	3	2	1	0
	(SP[7:0]) SPL							
Access	RW	RW	RW	RW	RW	RW	RW	RW
Reset	0	0	0	0	0	0	0	1

Bits 7:0 – (SP[7:0]) SPL: Stack Pointer Register
SPH and SPL are combined into SP. It means SPL[7:0] is SP[7:0].

AVR CPU Core – Pilha

- ✓ SP sempre aponta para o topo da pilha.

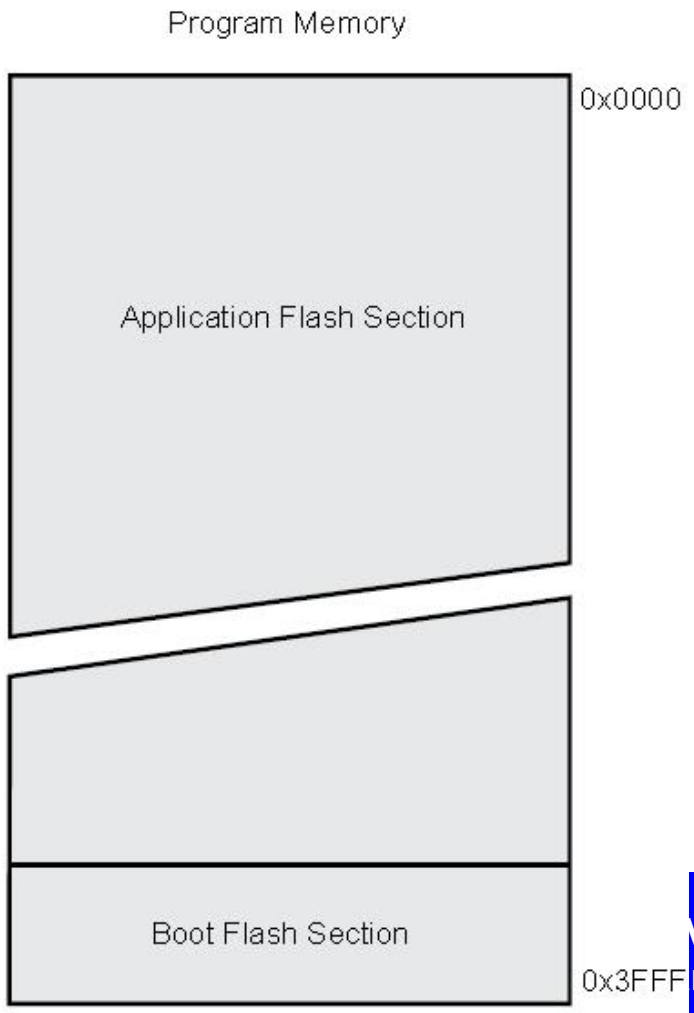
Instruction	Stack pointer	Description
PUSH	Decremented by 1	Data is pushed onto the stack
CALL ICALL RCALL	Decremented by 2	Return address is pushed onto the stack with a subroutine call or interrupt
POP	Incremented by 1	Data is popped from the stack
RET RETI	Incremented by 2	Return address is popped from the stack with return from subroutine or return from interrupt

Memórias

- ✓ Há dois principais espaços de memória: **programa** e **dados**. Além disso, há uma memória EEPROM para armazenamento de dados.
- ✓ **Memória de programa flash.**
  - 32 Kbytes – 16K x 16 (devido as instruções serem de 16 e 32 bits).
  - É dividida em duas seções:
    - boot loader e aplication program.

Memórias

- ✓ **Memória de programa flash.**

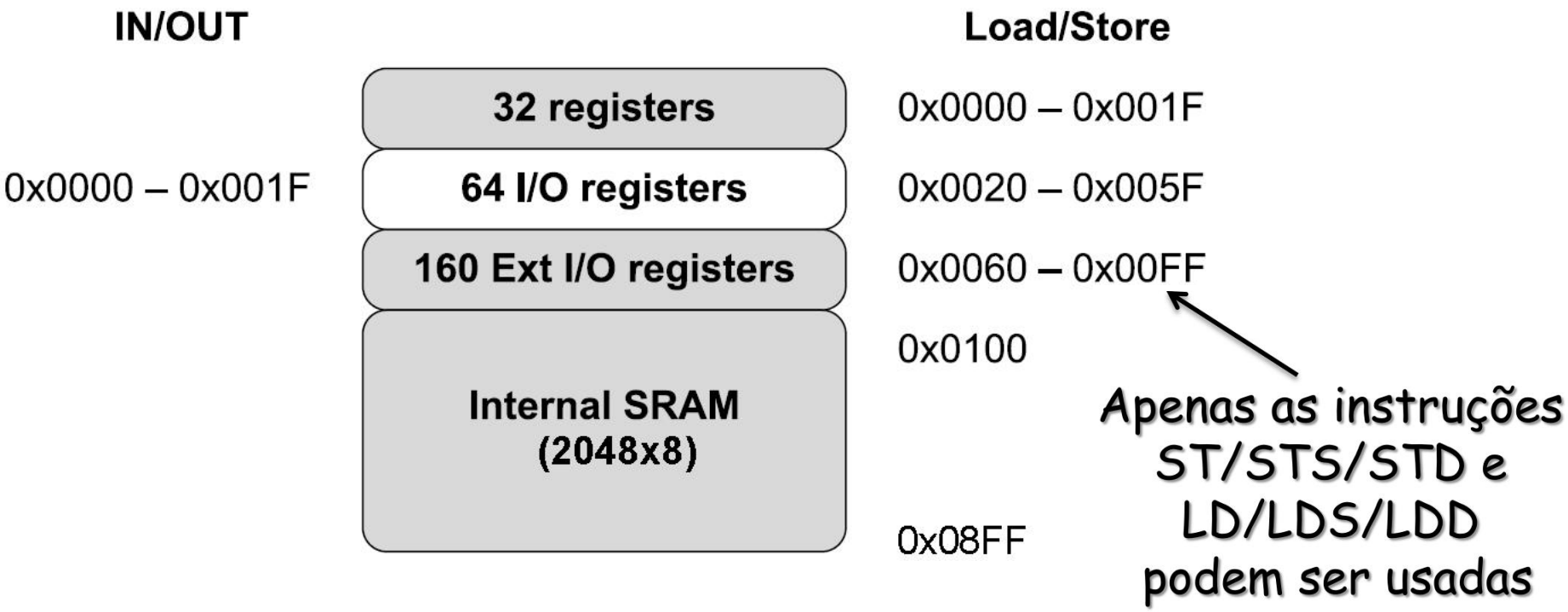


PC possui 14 bits.

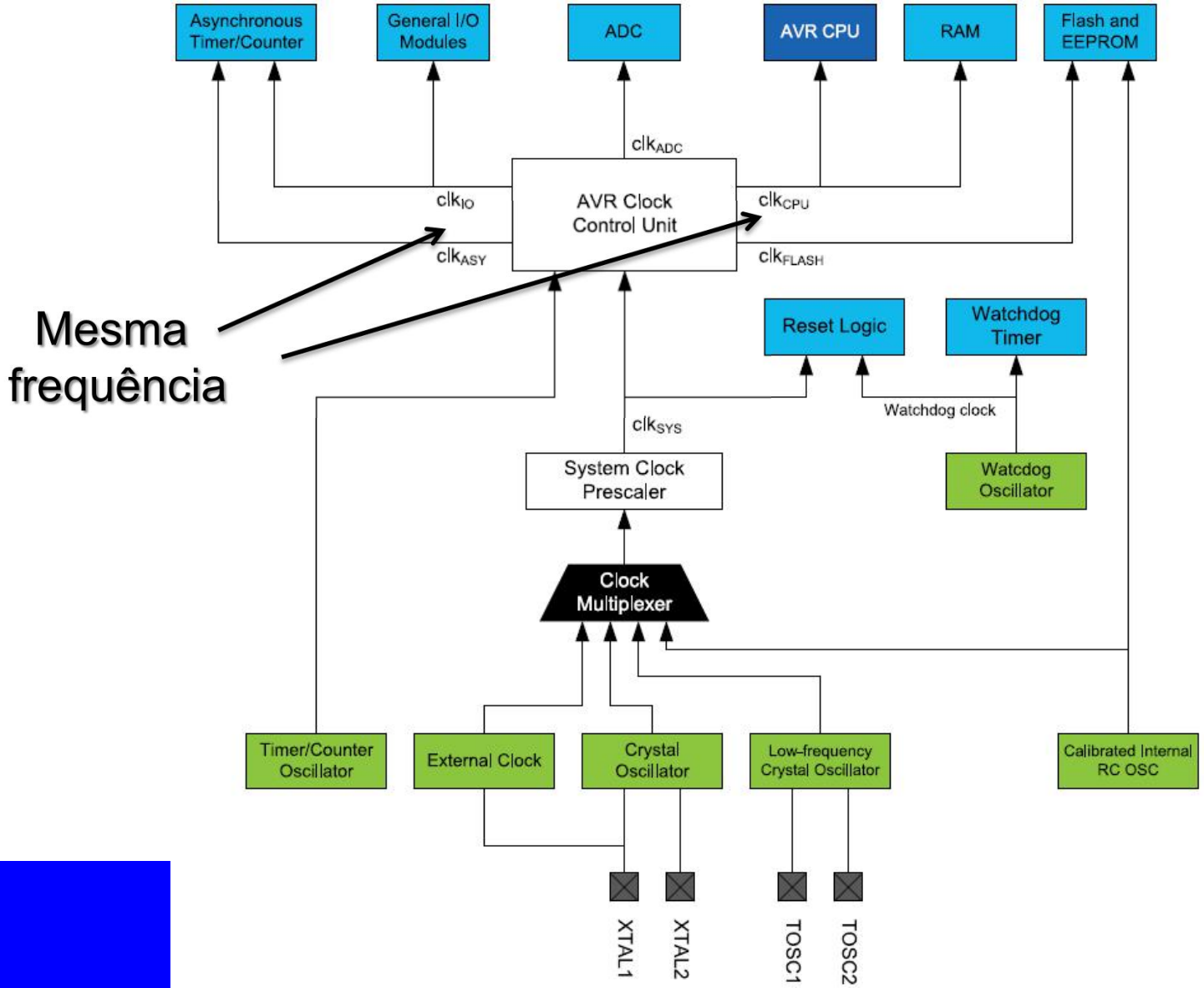
00 0000 0000 0000 (0000h)
11 1111 1111 1111 (3FFFh)

Memórias

- ✓ **Memória SRAM de dados.**



Clocks do Sistema e Opções de Clock



Clocks do Sistema e Opções de Clock

- ✓ Há **seis** opções de clock para o sistema **clkSYS**, selecionáveis via flash fuse (**CKSEL**).

Device Clocking Option	CKSEL[3:0]
Low Power Crystal Oscillator	1111 - 1000
Full Swing Crystal Oscillator	0111 - 0110
Low Frequency Crystal Oscillator	0101 - 0100
Internal 128kHz RC Oscillator	0011
Calibrated Internal RC Oscillator	0010
External Clock	0000
Reserved	0001

- ✓ Clock default (**calibrated internal RC oscilator** – em **8 MHz**) com o **fuse CKDIV8** selecionado, resultando em **1 MHz** de **clkSYS**.

Clocks do Sistema

- ✓ **Inicialização do clock.**
  - **Qualquer fonte de clock** - Vcc suficiente e um número mínimo de ciclos de oscilações (considerado estável).
  - **Garantir Vcc suficiente** → Reset interno com um atraso de time-out, marcado pelo oscilador do watchdog, com número de ciclos setado pelos bits fuses SUTx e CKSELx.

Clocks do Sistema

- ✓ **Inicialização do clock.**
  - Para garantir a estabilidade do clock, o reset é mantido por um número mínimo de ciclos de clock – oscilador start-up time – dependente do tipo de clock.
  - Por exemplo, no tipo "low frequency cristal", os ciclos variam de 6 a 32K.
  - O tempo de inicialização do clock = atraso time-out + start-up time.

Opções de Clock

- ✓ **Low Power Cristal Oscillator.**
- ✓ **Full Swing Cristal Oscillator.**
- ✓ Low Frequency Cristal Oscillator.
- ✓ **Calibrated Internal RC Oscillator.**
- ✓ **128 KHz Internal Oscillator.**
- ✓ **External Clock.**
- ✓ Timer/Counter Oscillator.

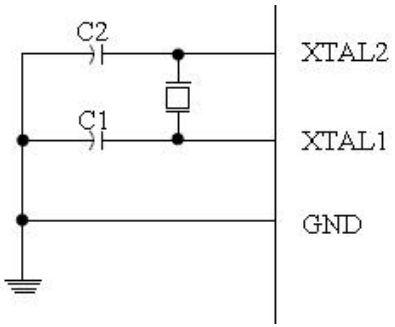
Opções de Clock

- ✓ **Low Power Cristal Oscillator.**
  - Baixo consumo de energia, não é capaz de fornecer para outras entradas de clock e é mais susceptível a ruídos em ambientes ruidosos.
  - Funciona em três diferentes modos.

Frequency Range [MHz]	CKSEL[3:1](2)	Range for Capacitors C1 and C2 [pF]
0.4 - 0.9	100(3)	–
0.9 - 3.0	101	12 - 22
3.0 - 8.0	110	12 - 22
8.0 - 16.0	111	12 - 22

Note:

1. If the crystal frequency exceeds the specification of the device (depends on V_{CC}), the CKDIV8 Fuse can be programmed in order to divide the internal frequency by 8. It must be ensured that the resulting divided clock meets the frequency specification of the device.
2. This is the recommended CKSEL settings for the difference frequency ranges.
3. This option should not be used with crystals, only with ceramic resonators.



Opções de Clock

- ✓ **Low Power Cristal Oscillator.**
  - Os fuses CKSEL0 e SUT[1:0] selecionam os start-up times.

Oscillator Source / Power Conditions	Start-up Time from Power-down and Power-save	Additional Delay from Reset (V_{CC} = 5.0V)	CKSEL0	SUT[1:0]
Ceramic resonator, fast rising power	258 CK	14CK + 4.1ms(1)	0	00
Ceramic resonator, slowly rising power	258 CK	14CK + 65ms(1)	0	01
Ceramic resonator, BOD enabled	1K CK	14CK(2)	0	10
Ceramic resonator, fast rising power	1K CK	14CK + 4.1ms(2)	0	11
Ceramic resonator, slowly rising power	1K CK	14CK + 65ms(2)	1	00
Crystal Oscillator, BOD enabled	16K CK	14CK	1	01
Crystal Oscillator, fast rising power	16K CK	14CK + 4.1ms	1	10
Crystal Oscillator, slowly rising power	16K CK	14CK + 65ms	1	11

Note:

1. These options should only be used when not operating close to the maximum frequency of the device, and only if frequency stability at start-up is not important for the application. These options are not suitable for crystals.
2. These options are intended for use with ceramic resonators and will ensure frequency stability at start-up. They can also be used with crystals when not operating close to the maximum frequency of the device, and if frequency stability at start-up is not important for the application.

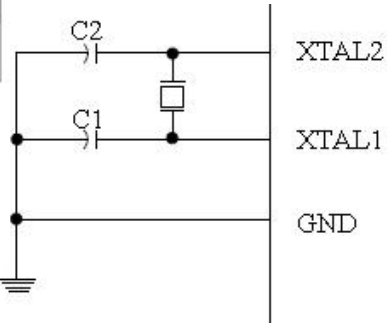
Opções de Clock

- ✓ **Full Swing Cristal Oscillator.**
  - Maior consumo de energia, é capaz de fornecer (pino XTAL2) para outras entradas de clock e em ambientes ruidosos.
  - Funciona somente com Vcc de 2,7 a 5,5 V.

Frequency Range[1] [MHz]	CKSEL[3:1]	Recommended Range for Capacitors C1 and C2 [pF]
0.4 - 20	011	12 - 22

Note:

1. If the crystal frequency exceeds the specification of the device (depends on V_{CC}), the CKDIV8 Fuse can be programmed in order to divide the internal frequency by 8. It must be ensured that the resulting divided clock meets the frequency specification of the device.



Opções de Clock

- ✓ **Full Swing Cristal Oscillator.**
  - Os fuses CKSEL0 e SUT[1:0] selecionam os start-up times.

Oscillator Source / Power Conditions	Start-Up Time from Power-down and Power-save	Additional Delay from Reset (V_{CC} = 5.0V)	CKSEL0	SUT[1:0]
Ceramic resonator, fast rising power	258 CK	14CK + 4.1ms(1)	0	00
Ceramic resonator, slowly rising power	258 CK	14CK + 65ms(1)	0	01
Ceramic resonator, BOD enabled	1K CK	14CK(2)	0	10
Ceramic resonator, fast rising power	1K CK	14CK + 4.1ms(2)	0	11
Ceramic resonator, slowly rising power	1K CK	14CK + 65ms(2)	1	00
Crystal Oscillator, BOD enabled	16K CK	14CK	1	01
Crystal Oscillator, fast rising power	16K CK	14CK + 4.1ms	1	10
Crystal Oscillator, slowly rising power	16K CK	14CK + 65ms	1	11

Note:

1. These options should only be used when not operating close to the maximum frequency of the device, and only if frequency stability at start-up is not important for the application. These options are not suitable for crystals.
2. These options are intended for use with ceramic resonators and will ensure frequency stability at start-up. They can also be used with crystals when not operating close to the maximum frequency of the device, and if frequency stability at start-up is not important for the application.

Opções de Clock

- ✓ **Calibrated Internal RC Oscillator.**
  - Por default, fornece um clock de 8 MHz com o fuse CKDIV8 programado.
  - Funciona sem componentes externos.

Opções de Clock

✓ Calibrated Internal RC Oscillator.

Frequency Range[1] [MHz]	CKSEL[3:0]
7.3 - 8.1	0010[2]

Note:

1. If 8MHz frequency exceeds the specification of the device (depends on V_{CC}), the CKDIV8 Fuse can be programmed in order to divide the internal frequency by 8.
2. The device is shipped with this option selected.

Power Conditions	Start-Up Time from Power-down and Power-Save	Additional Delay from Reset (V_{CC} = 5.0V)	SUT[1:0]
BOD enabled	6 CK	14CK	00
Fast rising power	6 CK	14CK + 4.1ms	01
Slowly rising power	6 CK	14CK + 65ms	10[1]
Reserved			11

Note:

1. The device is shipped with this option selected.

Opções de Clock

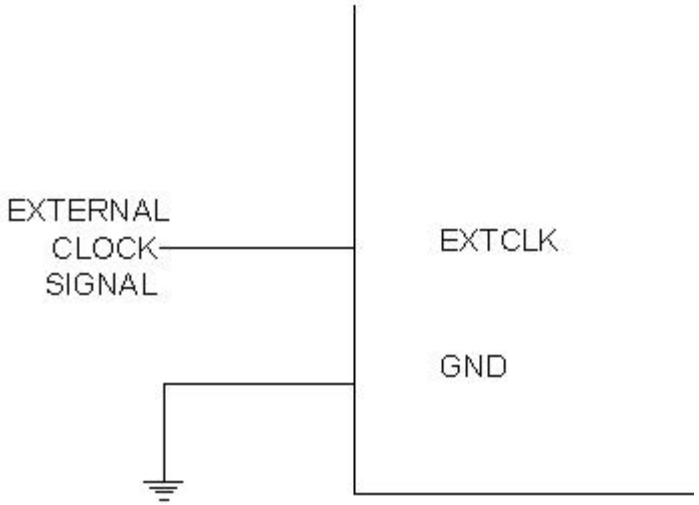
- ✓ **128 KHz Internal Oscillator.**
  - Frequência nominal de 128 KHz em 3 V e 25 °C.
  - Baixo consumo.

Nominal Frequency(1)	CKSEL[3:0]
128kHz	0011

Power Conditions	Start-Up Time from Power-down and Power-save	Additional Delay from Reset	SUT[1:0]
BOD enabled	6 CK	14CK	00
Fast rising power	6 CK	14CK + 4ms	01
Slowly rising power	6 CK	14CK + 64ms	10
Reserved			11

Opções de Clock

- ✓ **External Clock.**



Frequency	CKSEL[3:0]
0 - 20MHz	0000

Power Conditions	Start-Up Time from Power-down and Power-save	Additional Delay from Reset (V_{CC} = 5.0V)	SUT[1:0]
BOD enabled	6 CK	14CK	00
Fast rising power	6 CK	14CK + 4.1ms	01
Slowly rising power	6 CK	14CK + 65ms	10
Reserved			11

Opções de Clock

- ✓ **Buffer de saída do clock.**
  - O sinal do clock do sistema sai no **pino CLKO** (pino 14), se o fuse CKOUT for programado.
  - Utilizado para fornecer um clock para outros circuitos.

Opções de Clock

- ✓ **Prescaler do clock do sistema.**
  - Usado para diminuir a frequência de clock do sistema e o consumo de energia.
  - O clock do sistema é dividido por um fator configurado nos bits CLKPS [3:0] do registrador **CLKPR**.

Name: CLKPR
Offset: 0x61
Reset: Refer to the bit description
Property: -

Bit	7	6	5	4	3	2	1	0
	CLKPCE				CLKPS3	CLKPS2	CLKPS1	CLKPS0
Access	R/W				R/W	R/W	R/W	R/W
Reset	0				x	x	x	x

Opções de Clock

- ✓ **Prescaler do clock do sistema.**

Name: CLKPR
Offset: 0x61
Reset: Refer to the bit description
Property: -

Bit	7	6	5	4	3	2	1	0
	CLKPCE				CLKPS3	CLKPS2	CLKPS1	CLKPS0
Access	R/W				R/W	R/W	R/W	R/W
Reset	0				x	x	x	x

CLKPS[3:0]	Clock Division Factor
0000	1
0001	2
0010	4
0011	8
0100	16
0101	32
0110	64
0111	128
1000	256
1001	Reserved

Opções de Clock

- ✓ **Prescaler do clock do sistema.**

Name: CLKPR
Offset: 0x61
Reset: Refer to the bit description
Property: -

Bit	7	6	5	4	3	2	1	0
	CLKPCE				CLKPS3	CLKPS2	CLKPS1	CLKPS0
Access	R/W				R/W	R/W	R/W	R/W
Reset	0				x	x	x	x

- Bit 7 – CLKPCE: Deve ser '1' para habilitar a mudança dos bits CLKPS.
  - CLKPCE é atualizado somente se os outros bits (CLKPS) forem escritos com '0'.
  - CLKPCE é zerado automaticamente após 4 ciclos de clock ou quando CLKPS for escrito.

Opções de Clock

- ✓ **Prescaler do clock do sistema.**
  - Procedimento para mudança do prescaler.
    - CLKPR = 0x80.
    - Dentro de 4 ciclos de clock, escreva o valor desejado em CLKPS[3:0] e zere o bit CLKPCE.
      - CLKPR = 0x0N.

Programação Assembly

- ✓ Faça um programa que realiza a soma de dois números com 8 bits.
- ✓ Faça um programa que realiza a soma de dois números de 8 bits e se houver carry, armazene o resultado real da soma no registrador X (r27:r26).
- ✓ Faça um programa que realiza a soma de dois números de 16 bits.
- ✓ Faça um programa que resolve a seguinte equação: (a/2) + (b/2), sendo a e b números de 8 bits. se houver carry, armazene o resultado real no registrador X (r27:r26).

Programação Assembly

- ✓ Faça um programa em assembly que seta (0xFF) as posições da memória SRAM 0x0100 a 0x0120.
- ✓ Faça um programa em assembly que zera (0x00) as posições ímpares e seta (0xFF) as posições pares da memória SRAM 0x0100 a 0x0120.
- ✓ Faça um programa que implementa um delay de 1 us, 100 us, 1 ms, 100 ms e 1s.