

Curso: Engenharia de Computação

Arquitetura de Computadores

Prof. Clayton J A Silva, MSc
clayton.silva@professores.ibmec.edu.br



Microcontrolador Atmel ATmega V-2560

Manual do microcontrolador

- Arquitetura RISC
- 135 instruções
- Até 16 MIPS em 16 MHz
- *Flash 256 kB*
- *32 registradores de 8 bits*
- *4 kB EEPROM*
- *8 kB SRAM*
- *Periféricos*
 - *4 canais PWM de 8 bits*
 - *16 canais conversor AD*
 - ...
- *I/O*



Atmel ATmega640/V-1280/V-1281/V-2560/V-2561/V

Atmel ATmega640/V-1280/V-1281/V-2560/V-2561/V

8-bit Atmel Microcontroller with 16/32/64KB In-System Programmable Flash

DATASHEET

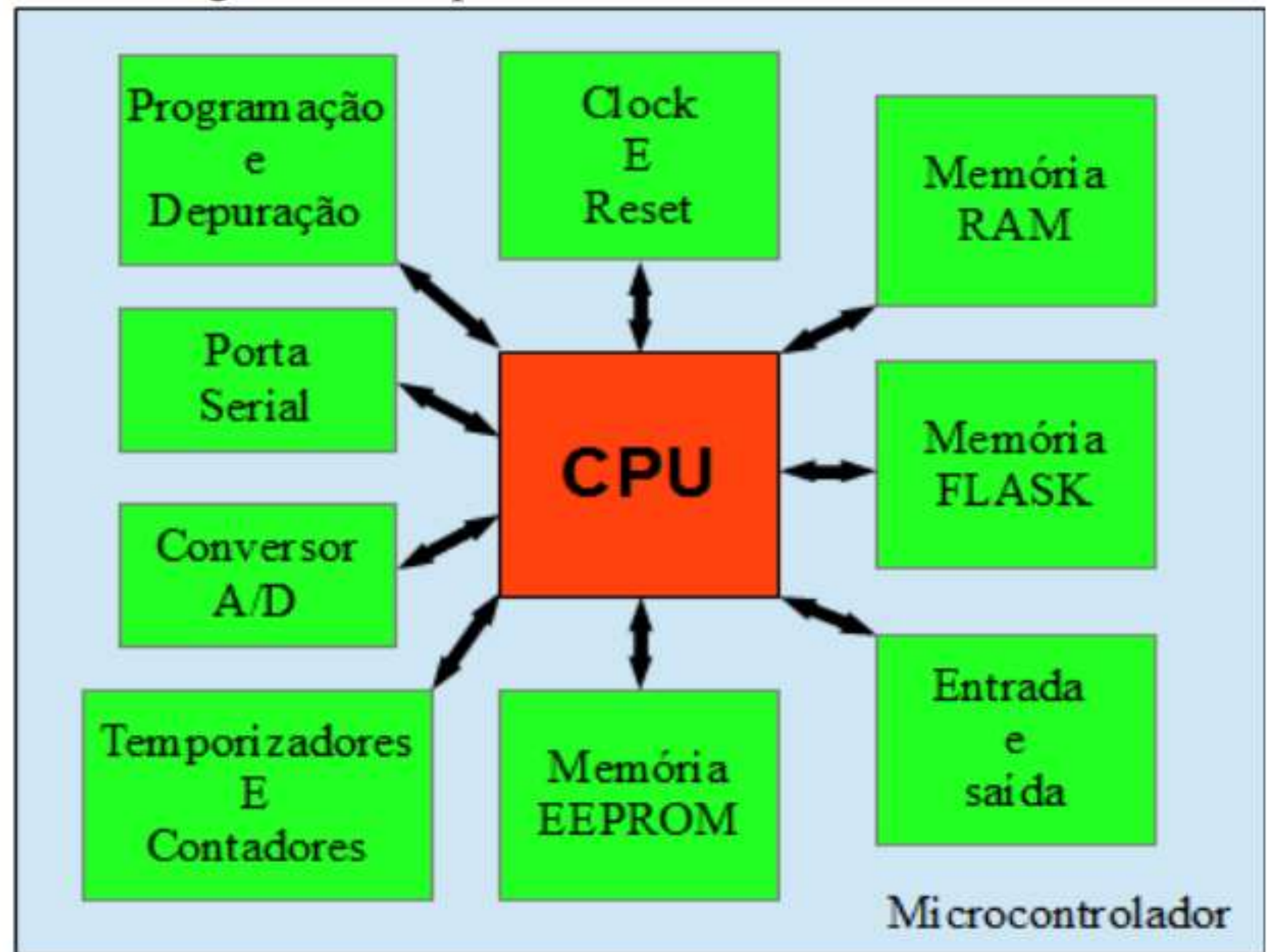
Features

- High Performance, Low Power Atmel® AVR® 8-Bit Microcontroller
- Advanced RISC Architecture
 - 135 Powerful Instructions – Most Single Clock Cycle Execution
 - 32 x 8 General Purpose Working Registers
 - Fully Static Operation
 - Up to 16 MIPS Throughput at 16MHz
 - On-Chip 2-cycle Multiplier
- High Endurance Non-volatile Memory Segments
 - 64K/128K/256KBytes of In-System Self-Programmable Flash
 - 4Kbytes EEPROM
 - 8Kbytes Internal SRAM
 - Write/Erase Cycles: 10,000 Flash/100,000 EEPROM
 - Data retention: 20 years at 85°C/ 100 years at 25°C
 - Optional Boot Code Section with Independent Lock Bits
 - In-System Programming by On-chip Boot Program
 - True Read-While-Write Operation
 - Programming Lock for Software Security
 - Endurance: Up to 64Kbytes Optional External Memory Space
- Atmel® QTouch® library support
 - Capacitive touch buttons, sliders and wheels
 - QTouch and QMatrix acquisition
 - Up to 64 sense channels
- JTAG (IEEE® std. 1149.1 compliant) Interface
 - Boundary-scan Capabilities According to the JTAG Standard
 - Extensive On-chip Debug Support
 - Programming of Flash, EEPROM, Fuses, and Lock Bits through the JTAG Interface
- Peripheral Features
 - Two 8-bit Timer/Counters with Separate Prescaler and Compare Mode
 - Four 16-bit Timer/Counter with Separate Prescaler, Compare- and Capture Mode
 - Real Time Counter with Separate Oscillator
 - Four 8-bit PWM Channels
 - Six/Twelve PWM Channels with Programmable Resolution from 2 to 16 Bits (ATmega1281/2561, ATmega640/1280/2560)
 - Output Compare Modulator
 - 8/16-channel, 10-bit ADC (ATmega1281/2561, ATmega640/1280/2560)
 - Two/Four Programmable Serial USART (ATmega1281/2561, ATmega640/1280/2560)
 - Master/Slave SPI Serial Interface
 - Byte Oriented 2-wire Serial Interface
 - Programmable Watchdog Timer with Separate On-chip Oscillator
 - On-chip Analog Comparator
 - Interrupt and Wake-up on Pin Change
- Special Microcontroller Features
 - Power-on Reset and Programmable Brown-out Detection
 - Internal Calibrated Oscillator
 - External and Internal Interrupt Sources
 - Six Sleep Modes: Idle, ADC Noise Reduction, Power-save, Power-down, Standby, and Extended Standby
- I/O and Packages
 - 54/86 Programmable I/O Lines (ATmega1281/2561, ATmega640/1280/2560)
 - 64-pad QFN/MLF, 64-lead TQFP (ATmega1281/2561)
 - 100-lead TQFP, 100-ball CBGA (ATmega640/1280/2560)
 - RoHS/Fully Green
- Temperature Range:
 - -40°C to 85°C Industrial
- Ultra-Low Power Consumption
 - Active Mode: 1MHz, 1.8V: 500µA
 - Power-down Mode: 0.1µA at 1.8V
- Speed Grade:
 - ATmega640V/ATmega1280V/ATmega1281V:
 - 0 - 4MHz @ 1.8V - 5.5V, 0 - 8MHz @ 2.7V - 5.5V
 - ATmega2560V/ATmega2561V:
 - 0 - 2MHz @ 1.8V - 5.5V, 0 - 8MHz @ 2.7V - 5.5V
 - ATmega640/ATmega1280/ATmega1281:
 - 0 - 8MHz @ 2.7V - 5.5V, 0 - 16MHz @ 4.5V - 5.5V
 - ATmega2560/ATmega2561:
 - 0 - 16MHz @ 4.5V - 5.5V

Microcontroladores

Circuitos integrados que possuem **internamente todos os componentes** necessário ao seu funcionamento (exceto fonte de alimentação).

Microcontroladores são computadores de um único chip.





Pinagem

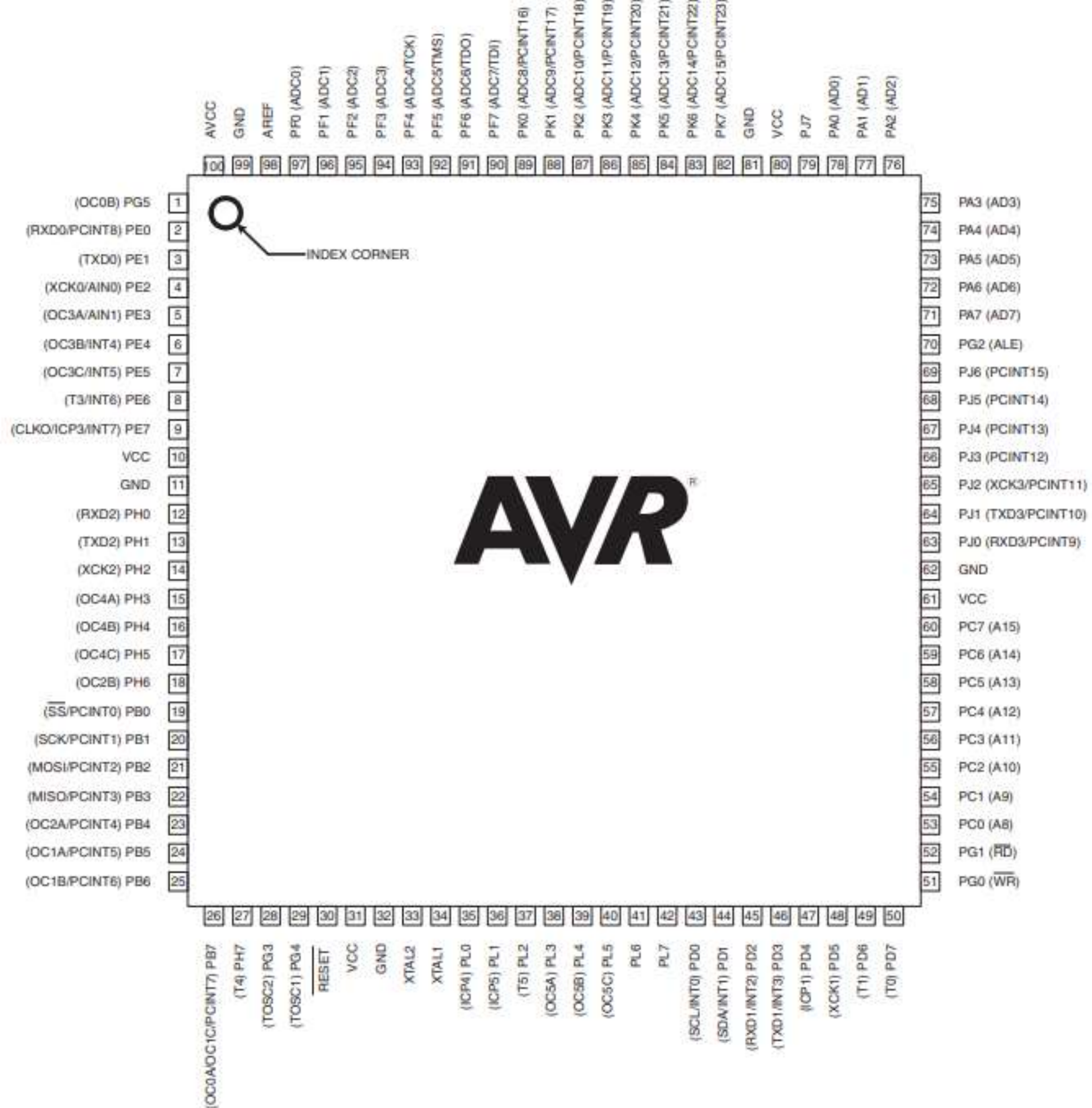
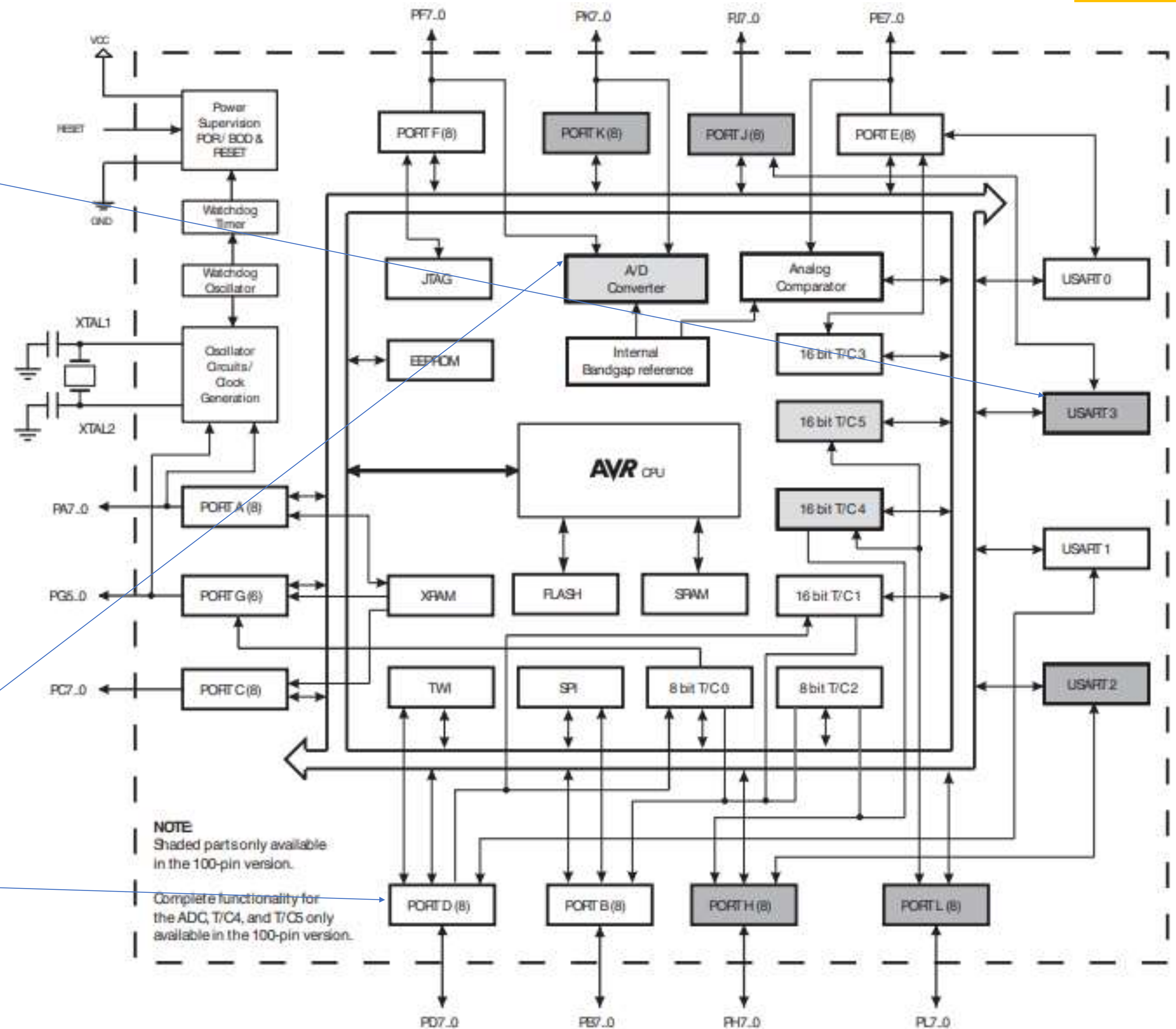


Diagrama em blocos

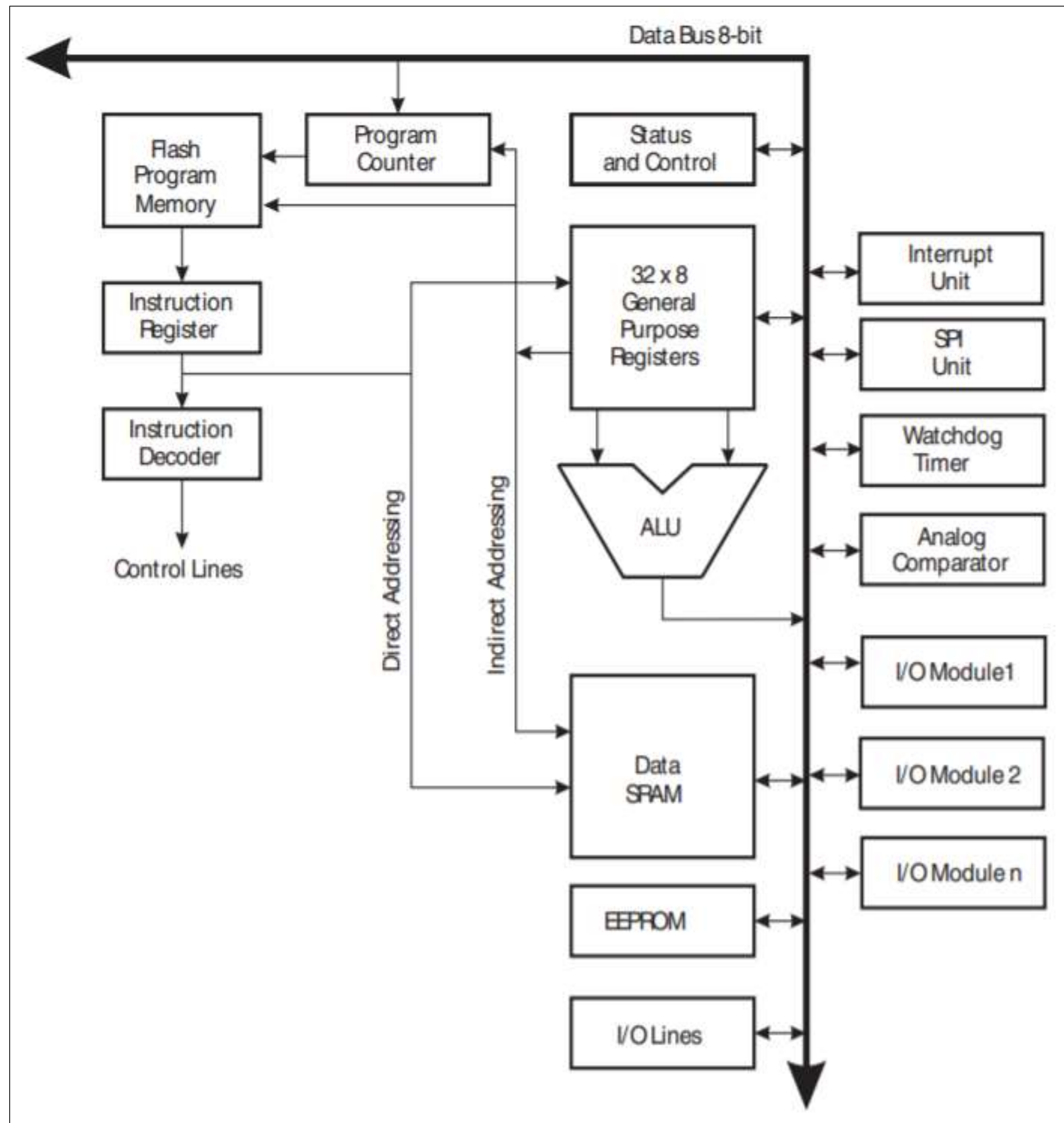
USART: receptor e transmissor síncrono e assíncrono universal.

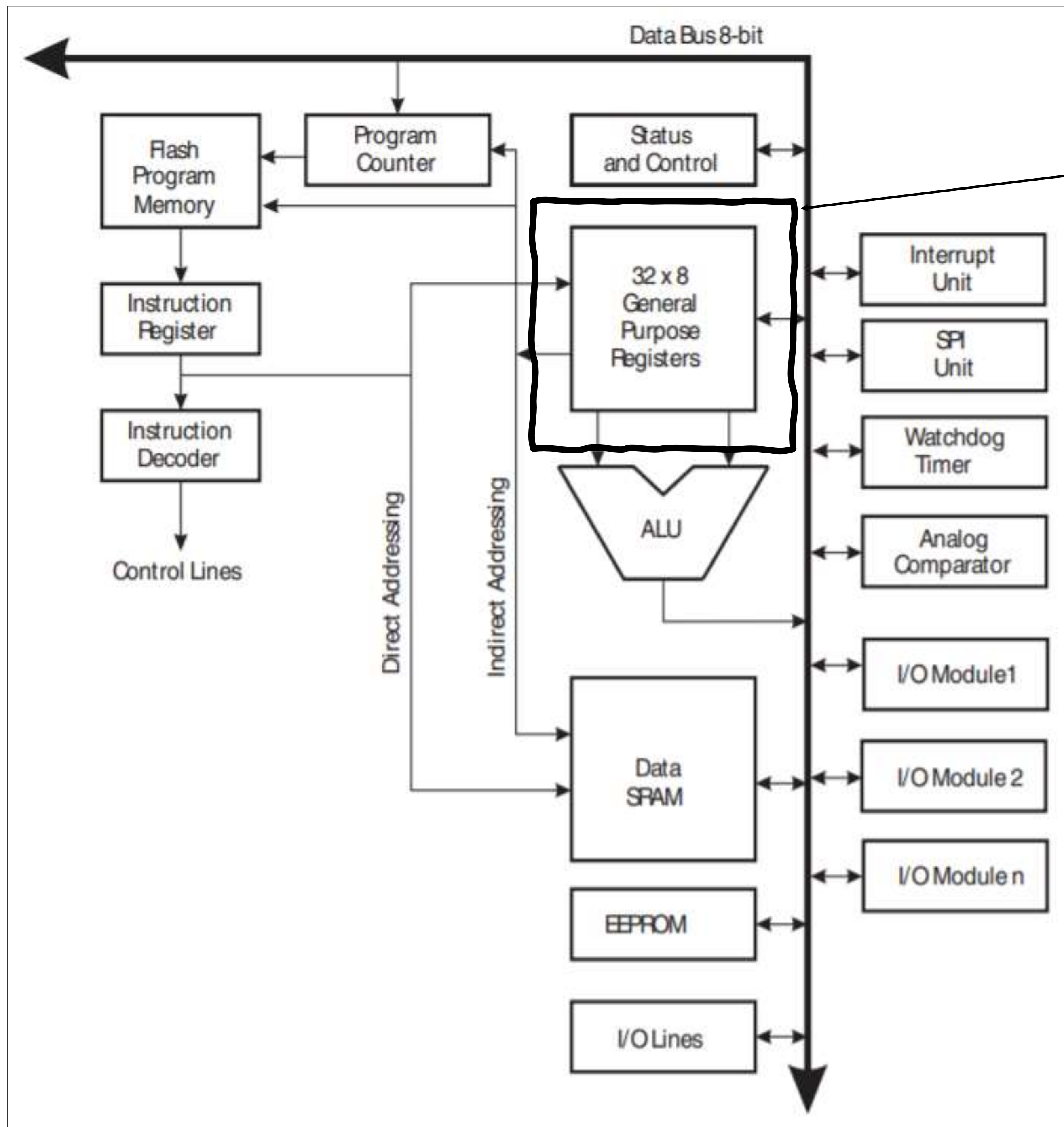
Conversor A/D

portas



Overview da arquitetura

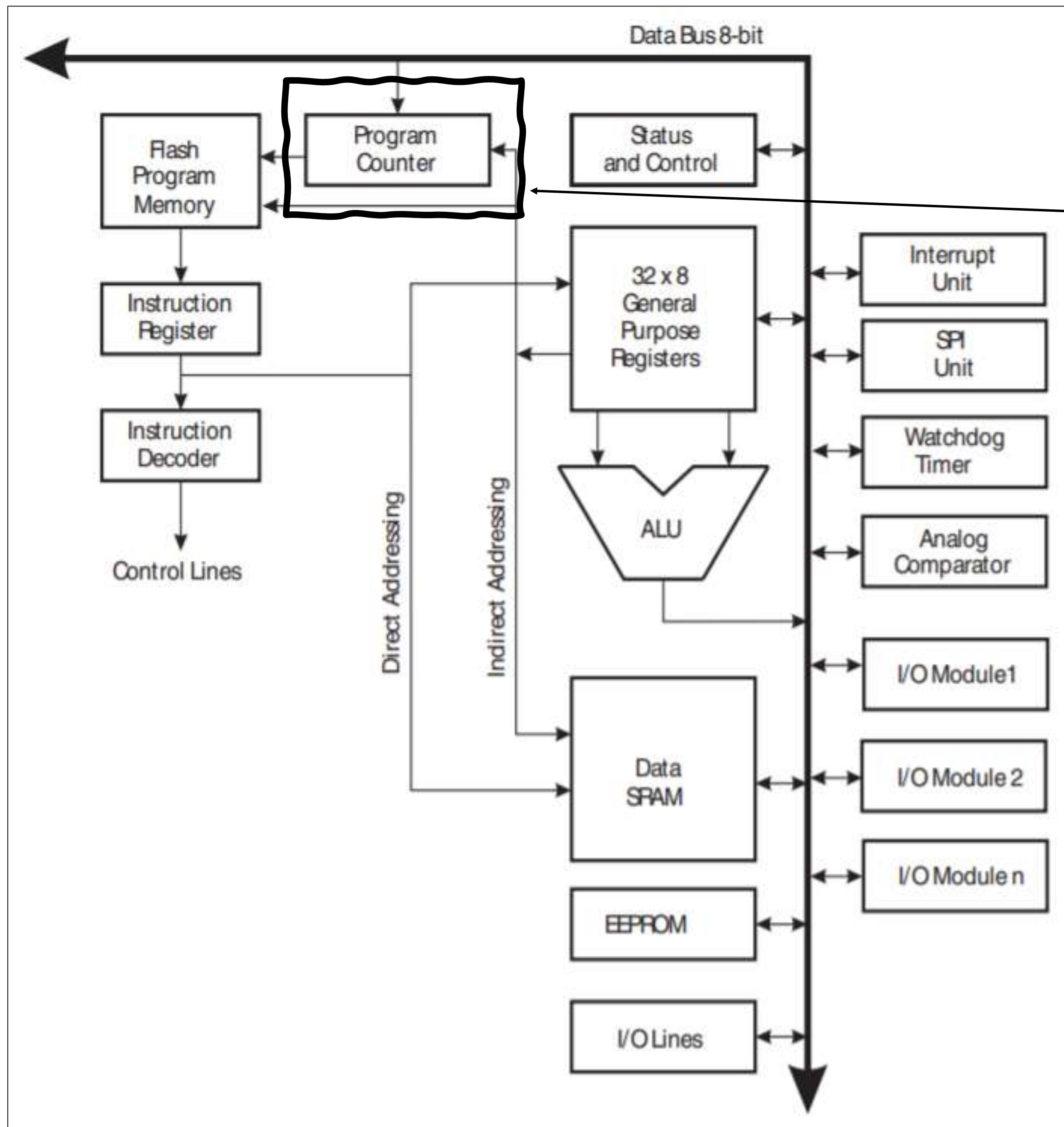




REGISTRADORES DE USO GERAL

32 (trinta e dois) registradores de uso geral com capacidade de armazenar **8 bits** cada, para uso geral e cujo **tempo de acesso é de um ciclo de clock**

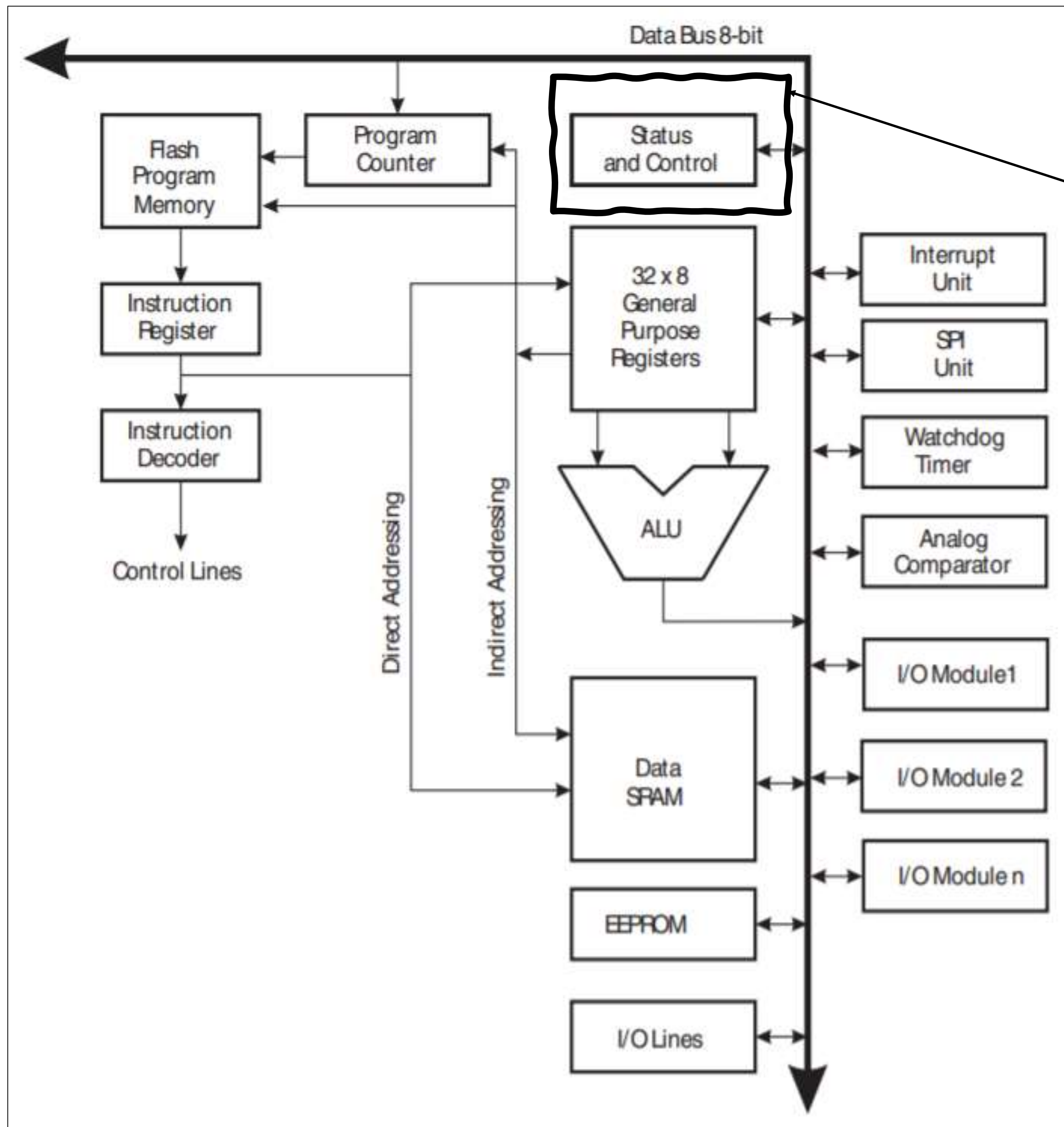
Sistema de memória



Contador de Programa (PC)

Registrador de **16-bits** que armazena o endereço da próxima instrução na FPM

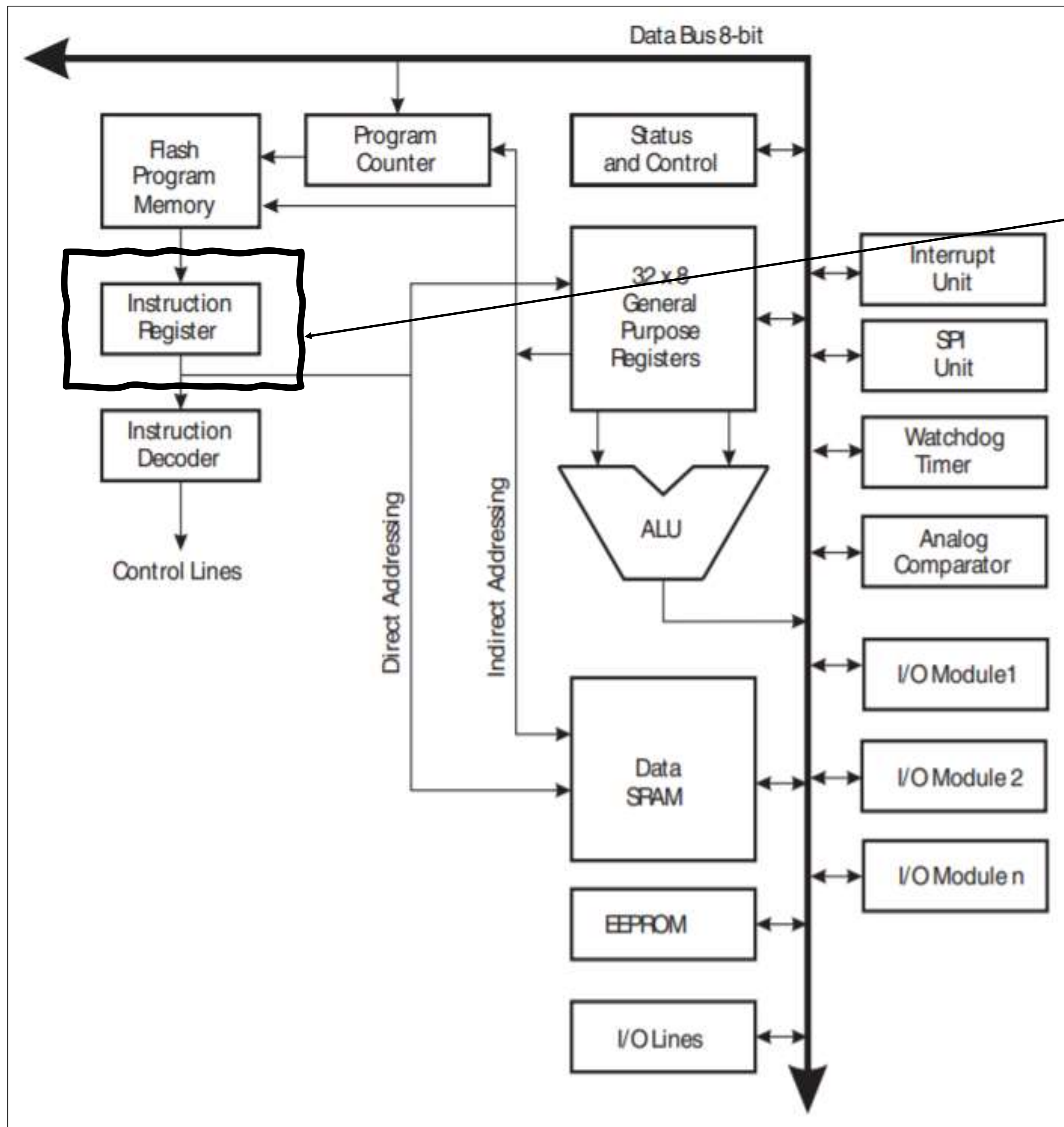
Sistema de memória



Registrador de Status

Registrador de **8-bits** que armazena o resultado da instrução aritmética mais recentemente executada

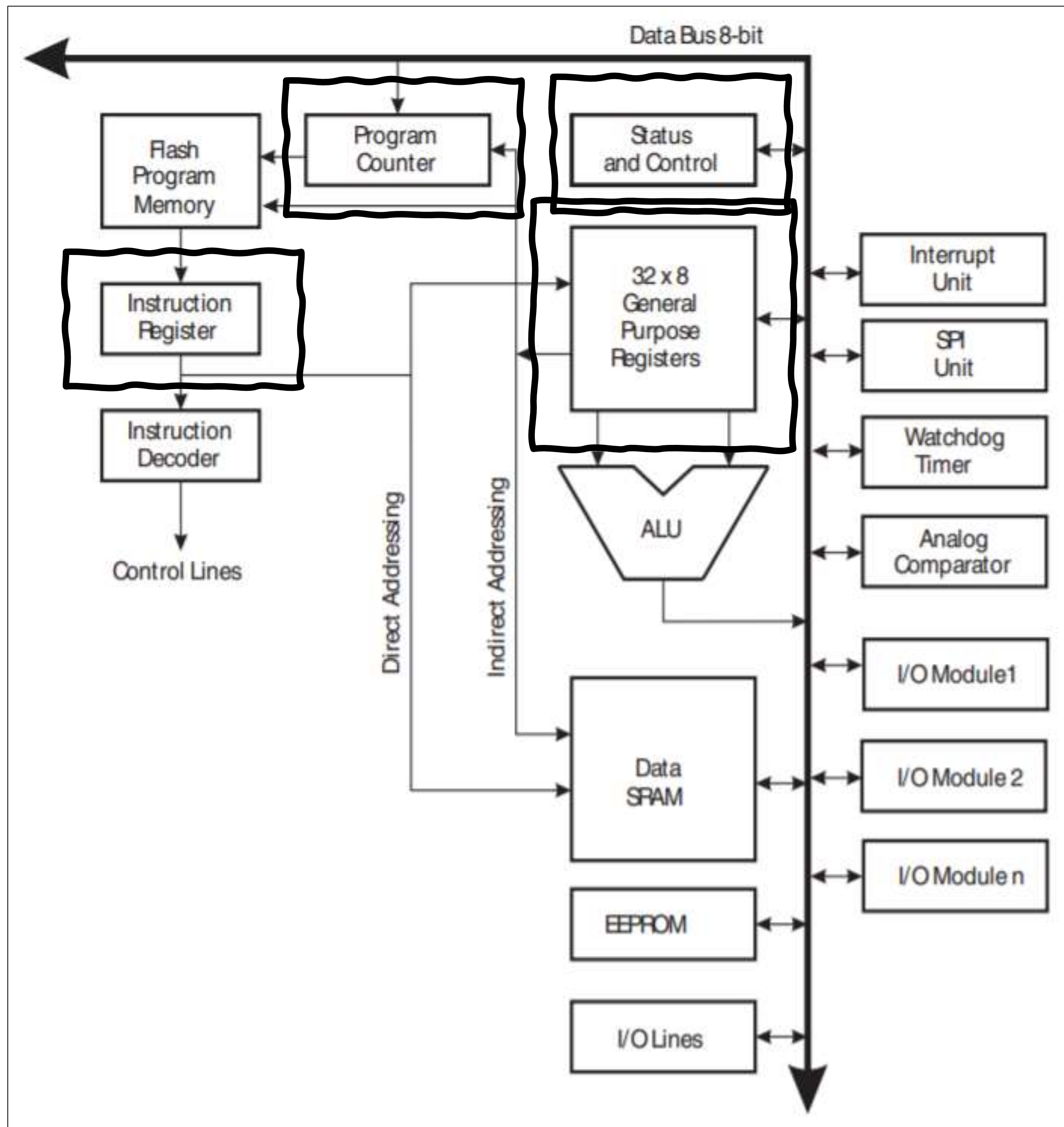
Sistema de memória



Registrador de Instrução

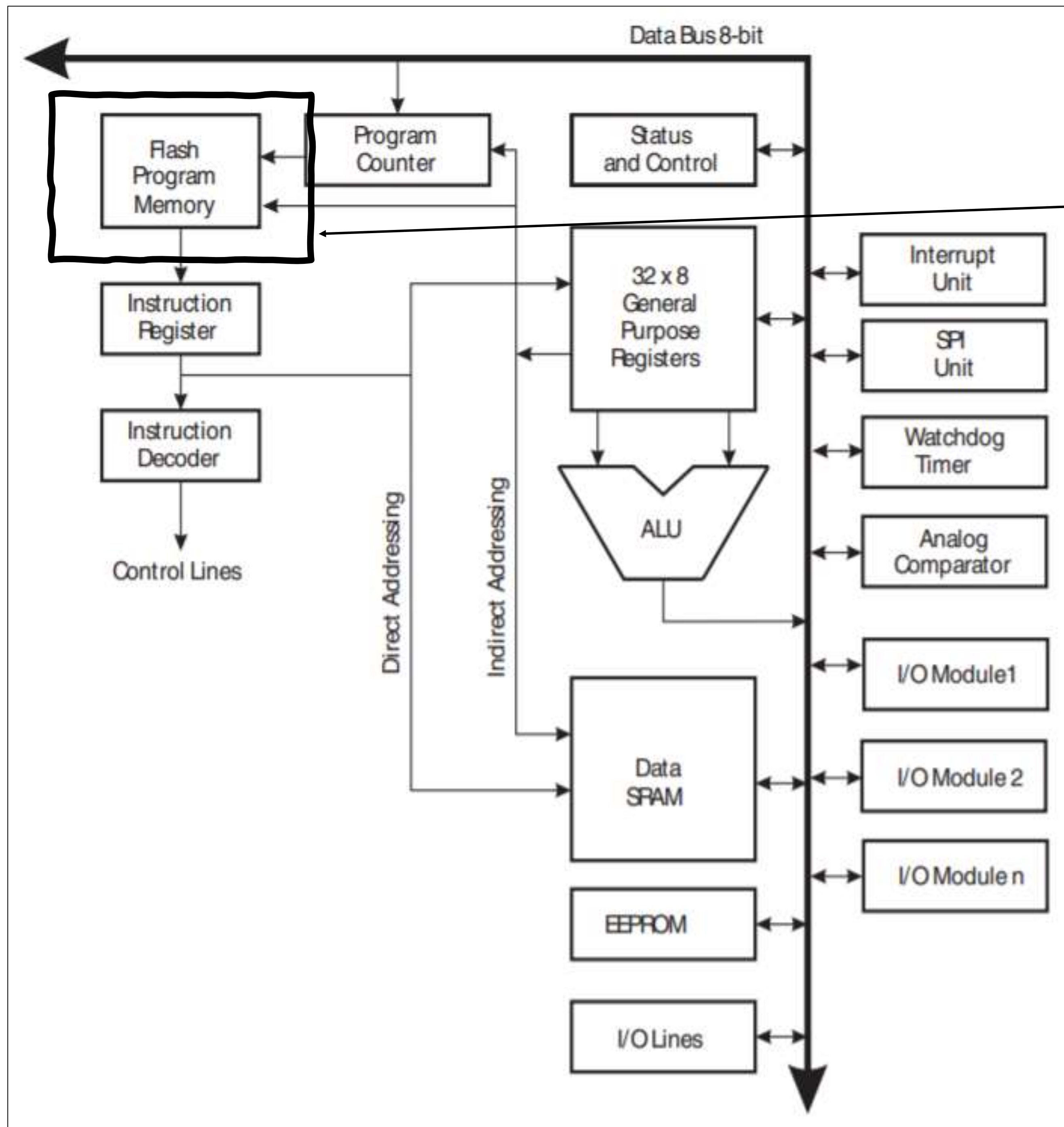
Registrador de **4-bits** que armazena o endereço da próxima instrução

Sistema de memória



Além desses registradores, o microcontrolador possui **64 registradores de I/O**.

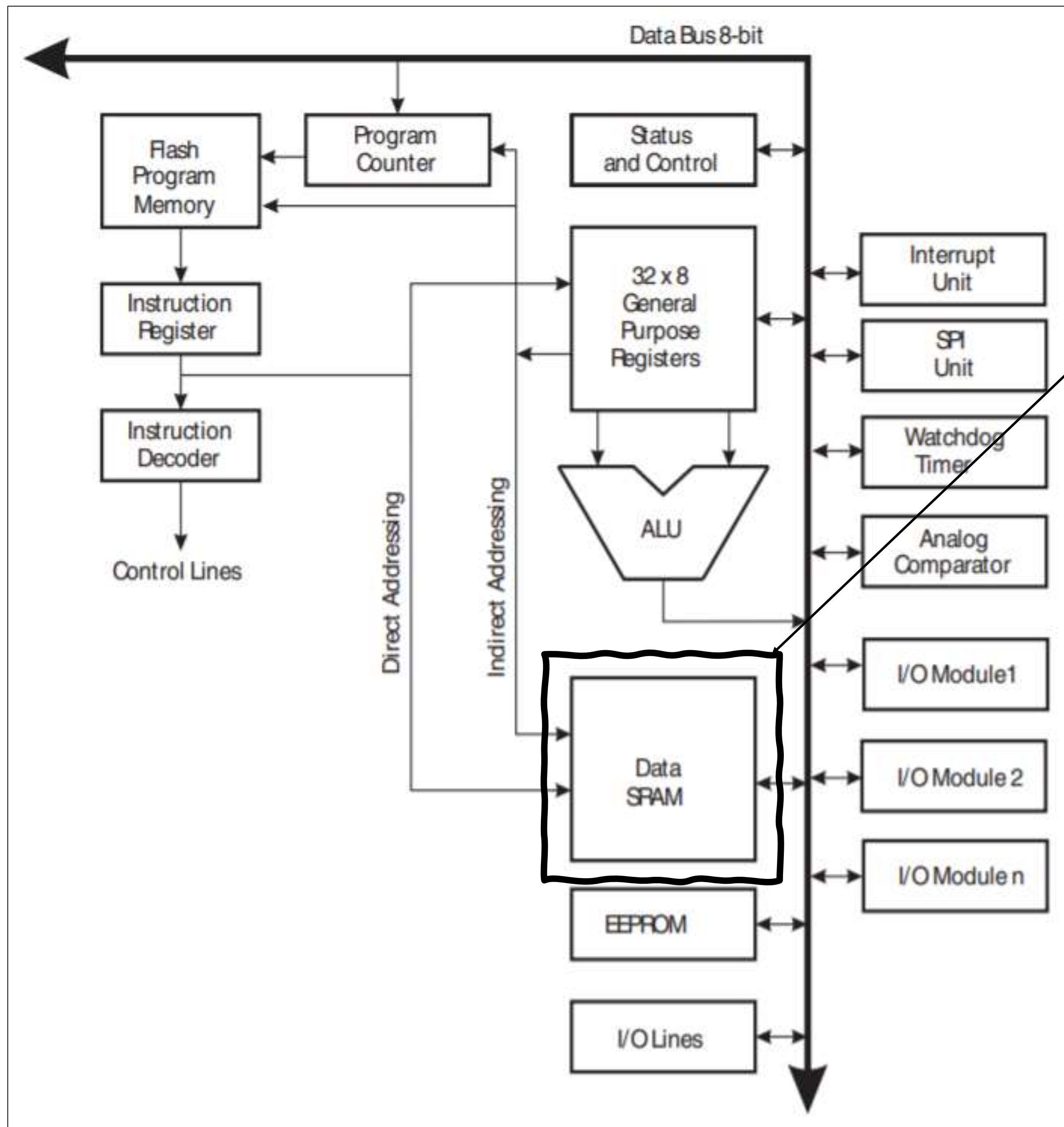
Sistema de memória



Memória FPM *Flash*

Memória Flash com capacidade de **256 kB**, palavras de **16-bits**, dividida em duas seções: a seção de ***boot*** do programa e a seção do **programa de aplicação**.

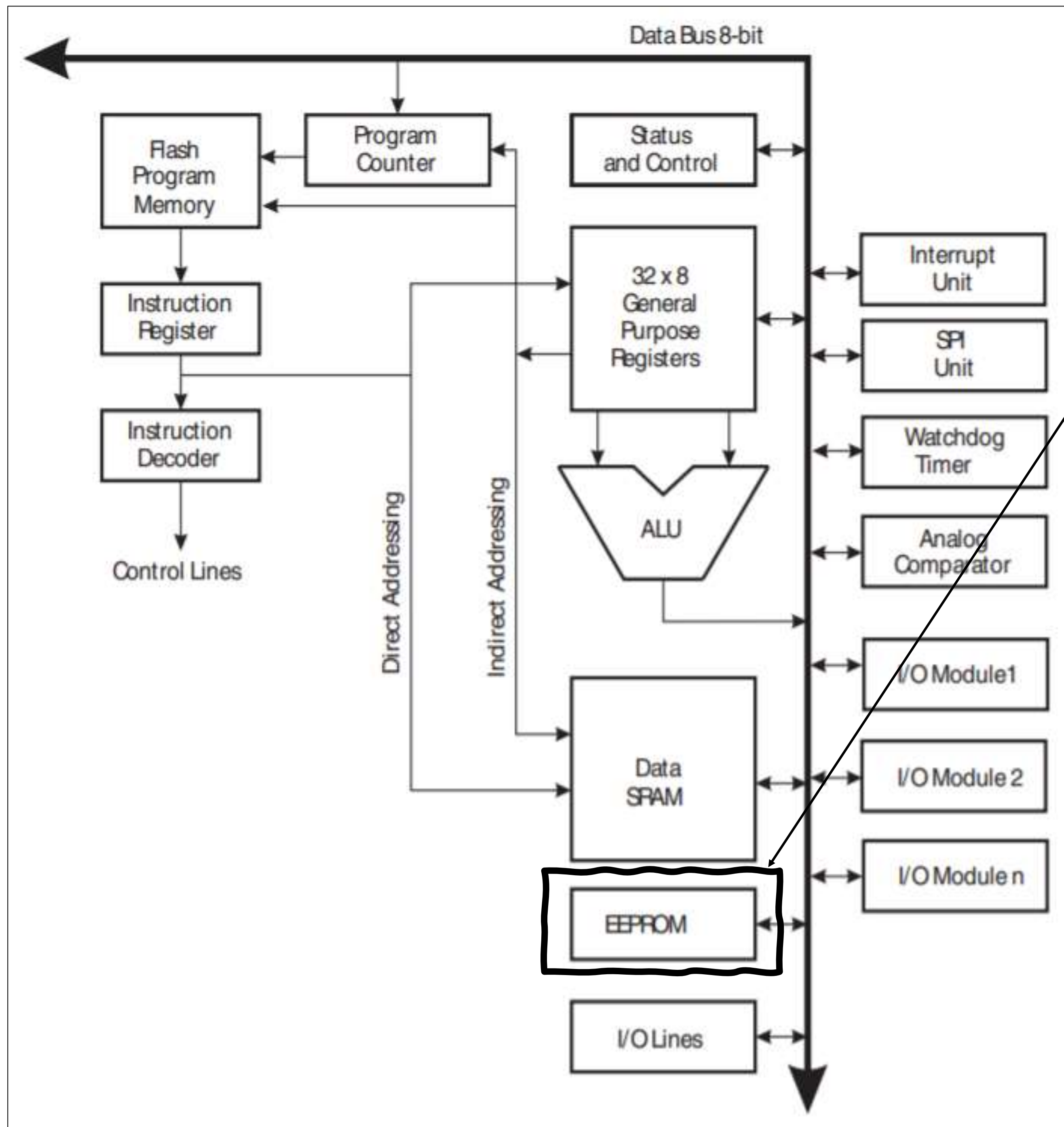
Sistema de memória



Memória SRAM

Memória com capacidade para **armazenamento de dados de 64 kB**

Sistema de memória



Memória EEPROM

Memória com capacidade para **armazenamento de dados de 4 Kb** para leitura e escrita de bytes

Sistema de memória

Endereçamento Atmel ATmega V-2560

Endereçamento *Flash* = *boot* + aplicação

Program Flash Memory Map

Address (HEX)

0

Application Flash Section

0x7FFF/0xFFFF/0x1FFFF

Boot Flash Section

Endereçamento da área de dados

Data Memory Map

Address (HEX)

0 - 1F

20 - 5F

60 - 1FF

200

21FF

2200

FFFF

32 Registers
64 I/O Registers
416 External I/O Registers
Internal SRAM (8192 × 8)
External SRAM (0 - 64K × 8)

Endereçamento de registradores de uso geral

Endereço									
R0	b7	b6	b5	b4	b3	b2	b1	b0	0x00
R1	b7	b6	b5	b4	b3	b2	b1	b0	0x01
R2	b7	b6	b5	b4	b3	b2	b1	b0	0x02
.....									
R31	b7	b6	b5	b4	b3	b2	b1	b0	0x1F

Registradores ponteiros X,Y,Z

Registrador X

R27

R26

Endereços

0x1B

0x1A

Registrador Y

R29

R28

Endereços

0x1D

0x1C

Registrador Z

R31

R30

Endereços

0x1F

0x1E

Registrador de *Status*

The AVR Status Register – SREG – is defined as:

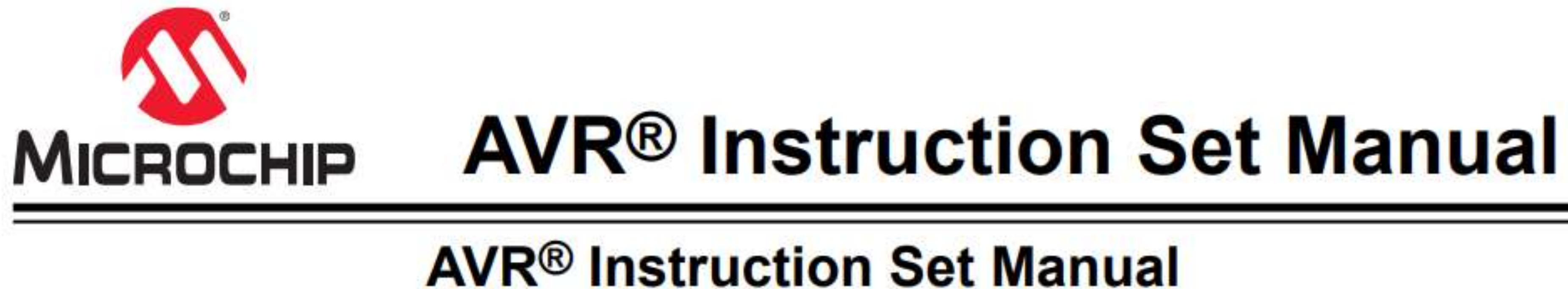
Bit	7	6	5	4	3	2	1	0	
0x3F (0x5F)	I	T	H	S	V	N	Z	C	SREG
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- Bit 2: N – Indica um resultado negativo em operação lógica e aritmética
- Bit 1: Z – Indica um resultado zero em operação lógica e aritmética
- Bit 0: C – Indica um *carry* em operação lógica e aritmética

Set de instruções Microcontrolador Atmel ATmega V-2560

Set de instruções

- Ver manual *AVR Instruction Set Manual*



Introduction

This manual gives an overview and explanation of every instruction available for 8-bit AVR® devices. Each instruction has its own section containing functional description, its opcode, and syntax, the end state of the status register, and cycle times.

The manual also contains an explanation of the different addressing modes used by AVR devices and an appendix listing all modern AVR devices and what instruction it has available.

Set de instruções

5.69 LDI – Load Immediate

5.69.1 Description

Loads an 8-bit constant directly to register 16 to 31.

Operation:

(i) $Rd \leftarrow K$

Syntax:

(i) LDI Rd,K

Operands:

$16 \leq d \leq 31, 0 \leq K \leq 255$

Program Counter:

$PC \leftarrow PC + 1$

16-bit Opcode:



5.69.2 Status Register (SREG) and Boolean Formula

I	T	H	S	V	N	Z	C
–	–	–	–	–	–	–	–

Set de instruções – Lógicas e aritméticas

Mnemonics	Operands	Description	Operation	Flags	#Clocks
ARITHMETIC AND LOGIC INSTRUCTIONS					
ADD	Rd, Rr	Add two Registers	$Rd \leftarrow Rd + Rr$	Z, C, N, V, H	1
ADC	Rd, Rr	Add with Carry two Registers	$Rd \leftarrow Rd + Rr + C$	Z, C, N, V, H	1
ADIW	Rdl,K	Add Immediate to Word	$Rdh:Rdl \leftarrow Rdh:Rdl + K$	Z, C, N, V, S	2
SUB	Rd, Rr	Subtract two Registers	$Rd \leftarrow Rd - Rr$	Z, C, N, V, H	1
SUBI	Rd, K	Subtract Constant from Register	$Rd \leftarrow Rd - K$	Z, C, N, V, H	1
SBC	Rd, Rr	Subtract with Carry two Registers	$Rd \leftarrow Rd - Rr - C$	Z, C, N, V, H	1
SBCI	Rd, K	Subtract with Carry Constant from Reg.	$Rd \leftarrow Rd - K - C$	Z, C, N, V, H	1
SBIW	Rdl,K	Subtract Immediate from Word	$Rdh:Rdl \leftarrow Rdh:Rdl - K$	Z, C, N, V, S	2
AND	Rd, Rr	Logical AND Registers	$Rd \leftarrow Rd \bullet Rr$	Z, N, V	1
ANDI	Rd, K	Logical AND Register and Constant	$Rd \leftarrow Rd \bullet K$	Z, N, V	1
OR	Rd, Rr	Logical OR Registers	$Rd \leftarrow Rd \vee Rr$	Z, N, V	1
ORI	Rd, K	Logical OR Register and Constant	$Rd \leftarrow Rd \vee K$	Z, N, V	1
EOR	Rd, Rr	Exclusive OR Registers	$Rd \leftarrow Rd \oplus Rr$	Z, N, V	1
COM	Rd	One's Complement	$Rd \leftarrow 0xFF - Rd$	Z, C, N, V	1
NEG	Rd	Two's Complement	$Rd \leftarrow 0x00 - Rd$	Z, C, N, V, H	1
SBR	Rd,K	Set Bit(s) in Register	$Rd \leftarrow Rd \vee K$	Z, N, V	1
CBR	Rd,K	Clear Bit(s) in Register	$Rd \leftarrow Rd \bullet (0xFF - K)$	Z, N, V	1
INC	Rd	Increment	$Rd \leftarrow Rd + 1$	Z, N, V	1
DEC	Rd	Decrement	$Rd \leftarrow Rd - 1$	Z, N, V	1
TST	Rd	Test for Zero or Minus	$Rd \leftarrow Rd \bullet Rd$	Z, N, V	1
CLR	Rd	Clear Register	$Rd \leftarrow Rd \oplus Rd$	Z, N, V	1
SER	Rd	Set Register	$Rd \leftarrow 0xFF$	None	1
MUL	Rd, Rr	Multiply Unsigned	$R1:R0 \leftarrow Rd \times Rr$	Z, C	2
MULS	Rd, Rr	Multiply Signed	$R1:R0 \leftarrow Rd \times Rr$	Z, C	2
MULSU	Rd, Rr	Multiply Signed with Unsigned	$R1:R0 \leftarrow Rd \times Rr$	Z, C	2
FMUL	Rd, Rr	Fractional Multiply Unsigned	$R1:R0 \leftarrow (Rd \times Rr) \ll 1$	Z, C	2
FMULS	Rd, Rr	Fractional Multiply Signed	$R1:R0 \leftarrow (Rd \times Rr) \ll 1$	Z, C	2
FMULSU	Rd, Rr	Fractional Multiply Signed with Unsigned	$R1:R0 \leftarrow (Rd \times Rr) \ll 1$	Z, C	2

Set de instruções – Deslocamento

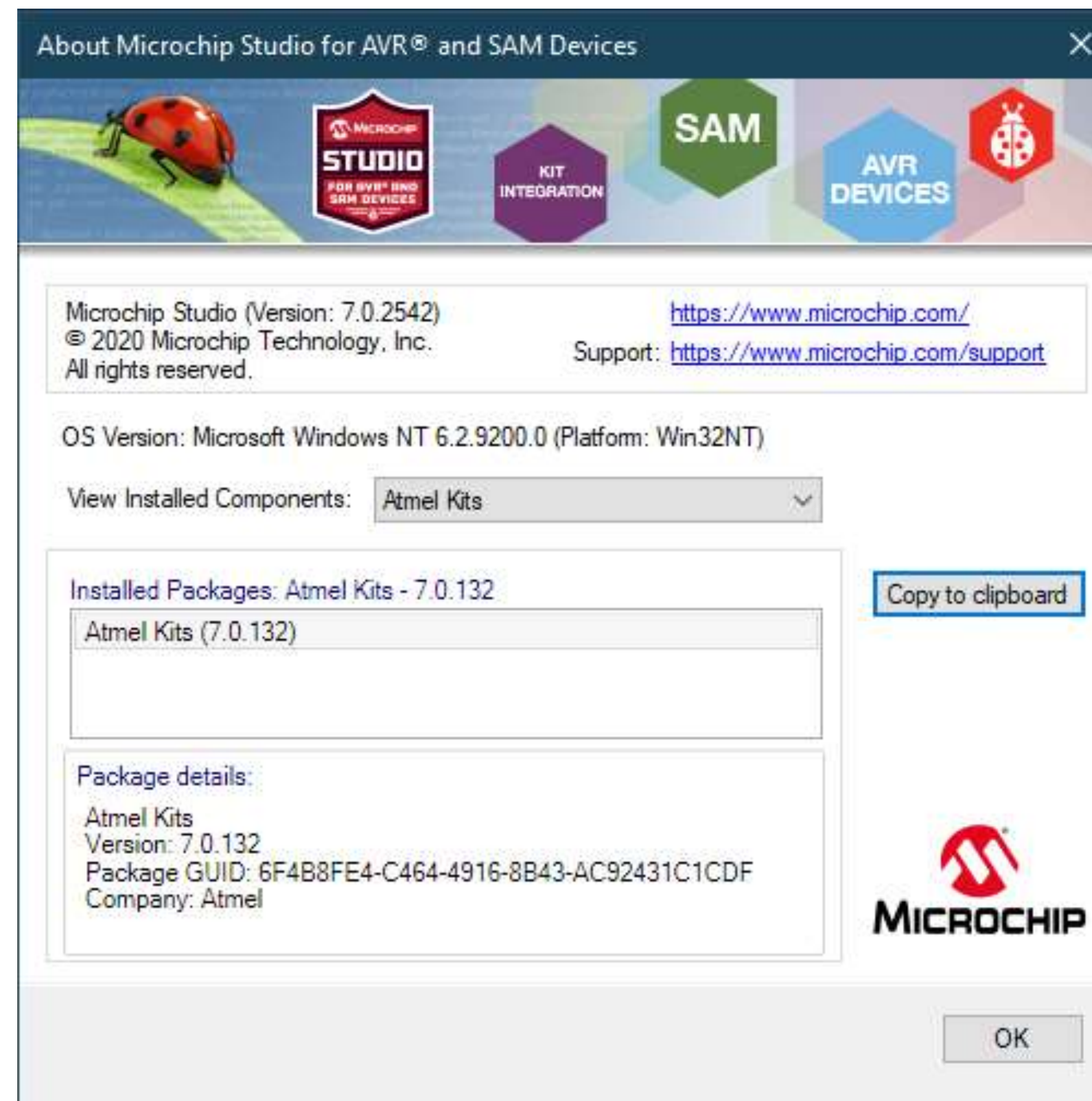
DATA TRANSFER INSTRUCTIONS					
MOV	Rd, Rr	Move Between Registers	$Rd \leftarrow Rr$	None	1
MOVW	Rd, Rr	Copy Register Word	$Rd+1:Rd \leftarrow Rr+1:Rr$	None	1
LDI	Rd, K	Load Immediate	$Rd \leftarrow K$	None	1
LD	Rd, X	Load Indirect	$Rd \leftarrow (X)$	None	2
LD	Rd, X+	Load Indirect and Post-Inc.	$Rd \leftarrow (X), X \leftarrow X + 1$	None	2
LD	Rd, -X	Load Indirect and Pre-Dec.	$X \leftarrow X - 1, Rd \leftarrow (X)$	None	2
LD	Rd, Y	Load Indirect	$Rd \leftarrow (Y)$	None	2
LD	Rd, Y+	Load Indirect and Post-Inc.	$Rd \leftarrow (Y), Y \leftarrow Y + 1$	None	2
LD	Rd, -Y	Load Indirect and Pre-Dec.	$Y \leftarrow Y - 1, Rd \leftarrow (Y)$	None	2
LDD	Rd, Y+q	Load Indirect with Displacement	$Rd \leftarrow (Y + q)$	None	2
LD	Rd, Z	Load Indirect	$Rd \leftarrow (Z)$	None	2
LD	Rd, Z+	Load Indirect and Post-Inc.	$Rd \leftarrow (Z), Z \leftarrow Z + 1$	None	2
LD	Rd, -Z	Load Indirect and Pre-Dec.	$Z \leftarrow Z - 1, Rd \leftarrow (Z)$	None	2
LDD	Rd, Z+q	Load Indirect with Displacement	$Rd \leftarrow (Z + q)$	None	2
LDS	Rd, k	Load Direct from SRAM	$Rd \leftarrow (k)$	None	2
ST	X, Rr	Store Indirect	$(X) \leftarrow Rr$	None	2
ST	X+, Rr	Store Indirect and Post-Inc.	$(X) \leftarrow Rr, X \leftarrow X + 1$	None	2
ST	-X, Rr	Store Indirect and Pre-Dec.	$X \leftarrow X - 1, (X) \leftarrow Rr$	None	2
ST	Y, Rr	Store Indirect	$(Y) \leftarrow Rr$	None	2
ST	Y+, Rr	Store Indirect and Post-Inc.	$(Y) \leftarrow Rr, Y \leftarrow Y + 1$	None	2
ST	-Y, Rr	Store Indirect and Pre-Dec.	$Y \leftarrow Y - 1, (Y) \leftarrow Rr$	None	2
STD	Y+q, Rr	Store Indirect with Displacement	$(Y + q) \leftarrow Rr$	None	2
ST	Z, Rr	Store Indirect	$(Z) \leftarrow Rr$	None	2
ST	Z+, Rr	Store Indirect and Post-Inc.	$(Z) \leftarrow Rr, Z \leftarrow Z + 1$	None	2
ST	-Z, Rr	Store Indirect and Pre-Dec.	$Z \leftarrow Z - 1, (Z) \leftarrow Rr$	None	2
STD	Z+q, Rr	Store Indirect with Displacement	$(Z + q) \leftarrow Rr$	None	2
STS	k, Rr	Store Direct to SRAM	$(k) \leftarrow Rr$	None	2
LPM		Load Program Memory	$R0 \leftarrow (Z)$	None	3
LPM	Rd, Z	Load Program Memory	$Rd \leftarrow (Z)$	None	3

Set de instruções – Deslocamento

DATA TRANSFER INSTRUCTIONS					
LPM	Rd, Z+	Load Program Memory and Post-Inc	$Rd \leftarrow (Z), Z \leftarrow Z+1$	None	3
ELPM		Extended Load Program Memory	$R0 \leftarrow (RAMPZ:Z)$	None	3
ELPM	Rd, Z	Extended Load Program Memory	$Rd \leftarrow (RAMPZ:Z)$	None	3
ELPM	Rd, Z+	Extended Load Program Memory	$Rd \leftarrow (RAMPZ:Z), RAMPZ:Z \leftarrow RAMPZ:Z+1$	None	3
SPM		Store Program Memory	$(Z) \leftarrow R1:R0$	None	-
IN	Rd, P	In Port	$Rd \leftarrow P$	None	1
OUT	P, Rr	Out Port	$P \leftarrow Rr$	None	1
PUSH	Rr	Push Register on Stack	$STACK \leftarrow Rr$	None	2
POP	Rd	Pop Register from Stack	$Rd \leftarrow STACK$	None	2

Microchip Studio 7

Manual IDE



Ver manual AVR *Assembler*



AVR Assembler

AVR Assembler

Preface

Welcome to the Microchip AVR[®] Assembler.

The Assembler generates fixed code allocations, consequently no linking is necessary.

The AVR Assembler is the assembler formerly known as AVR Assembler 2 (AVRASM2). The former AVRASM distributed with AVR Studio[®] 4 has now been obsoleted and will not be distributed with current products.

For documentation on the instruction set of the AVR family of microcontrollers, refer to the [8-bit AVR Instruction Set Manual](#).

Ver manual da IDE



Atmel Studio 7 User Guide

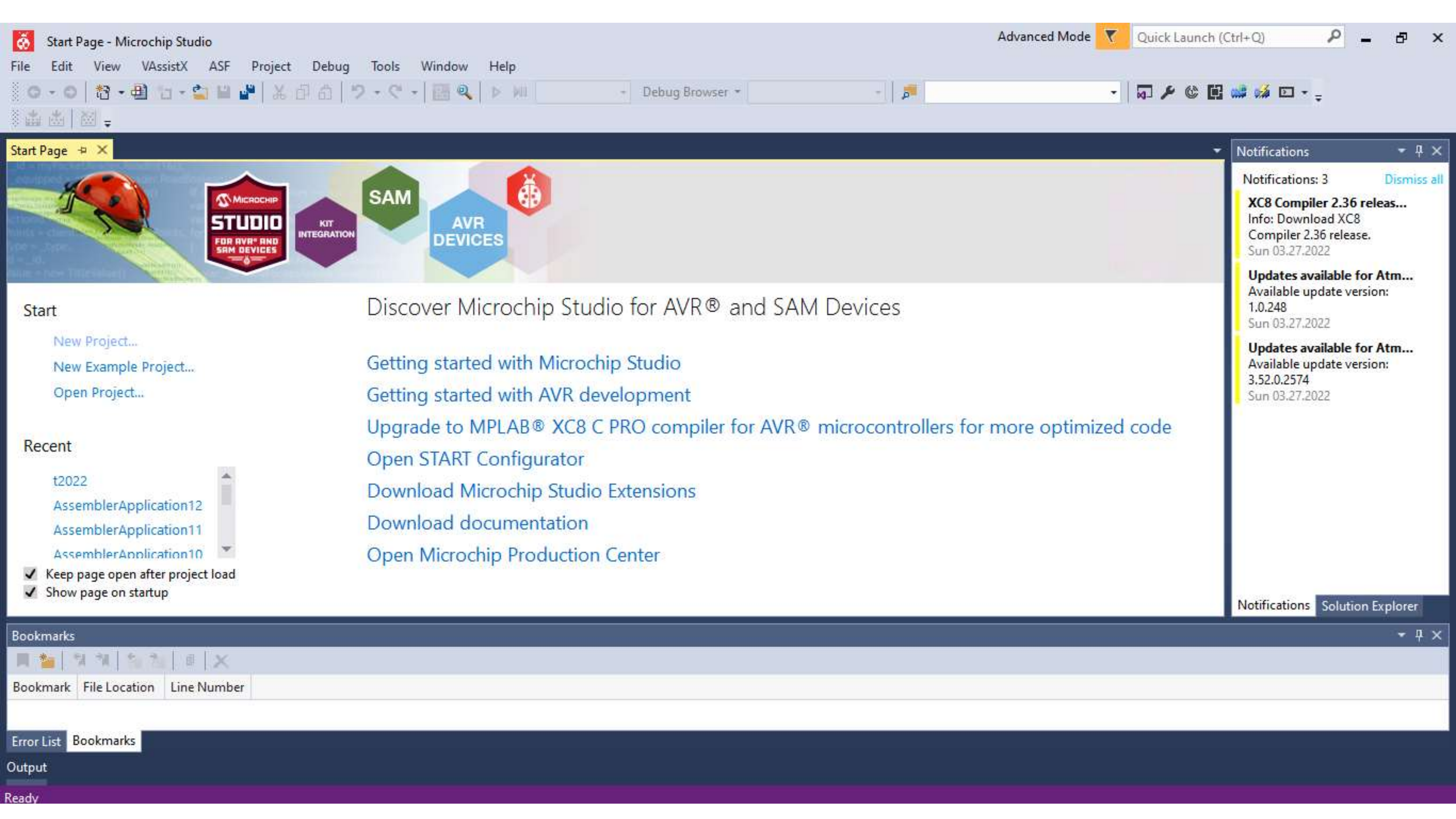
Atmel Studio 7

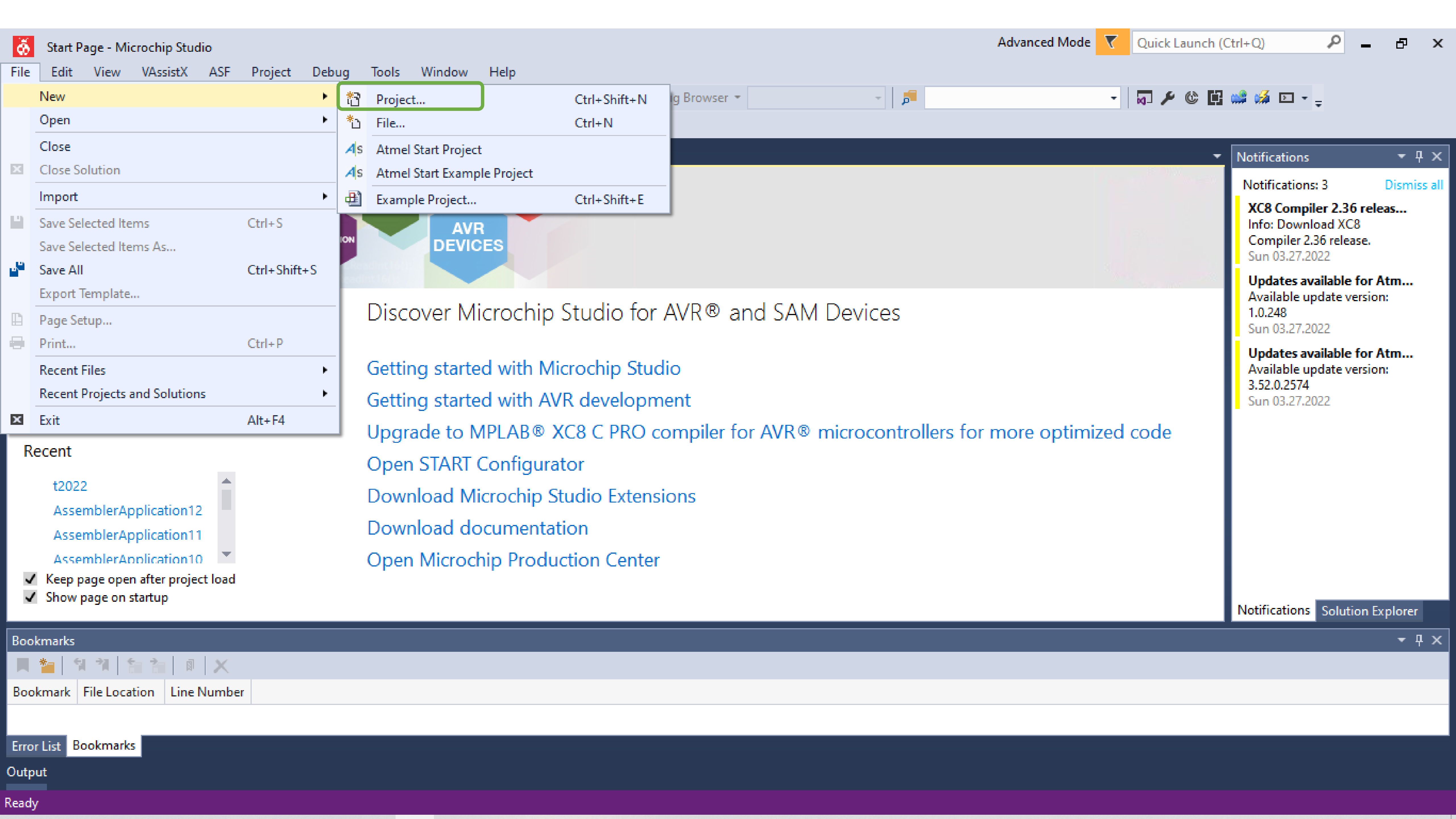
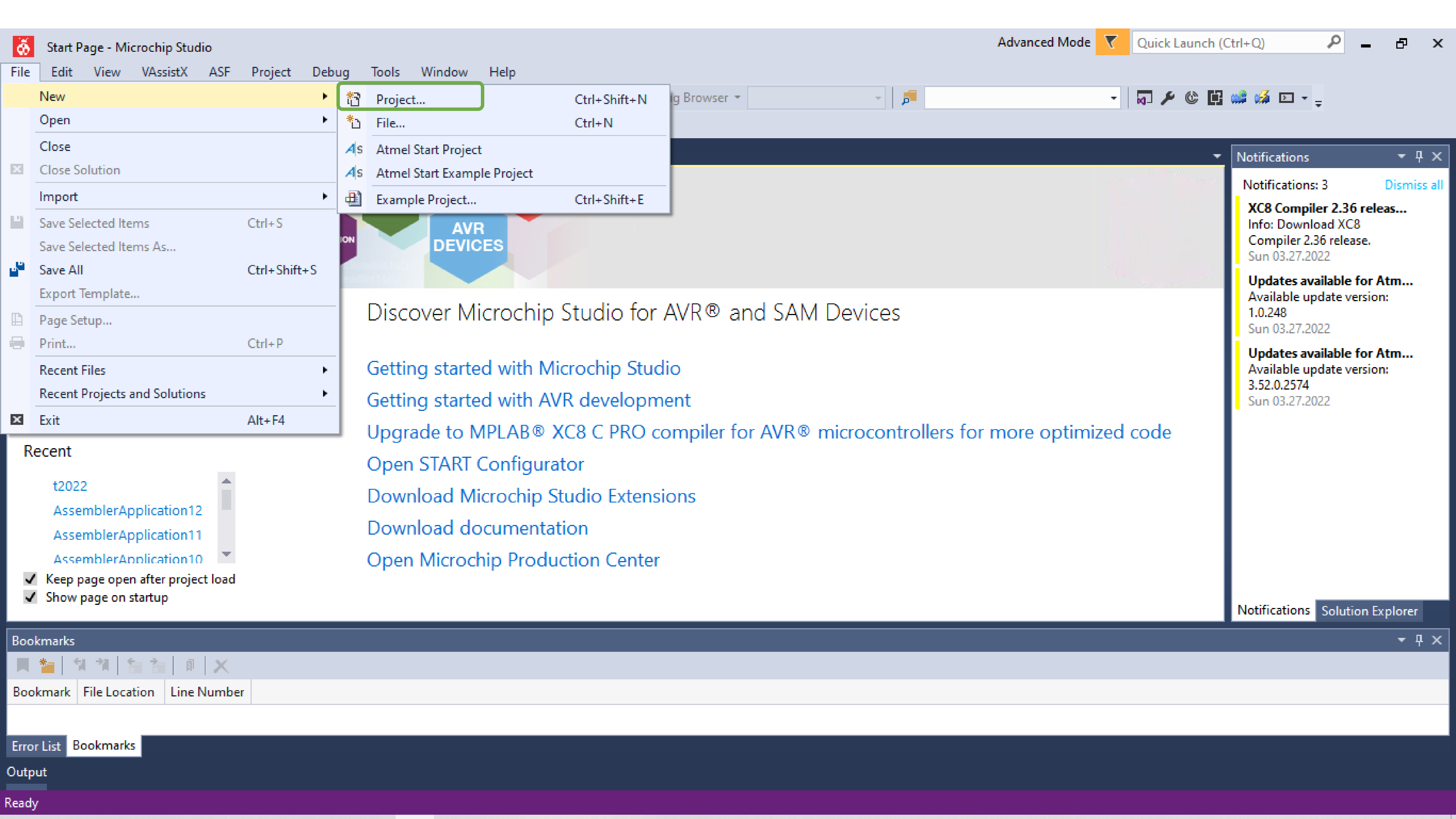
Preface

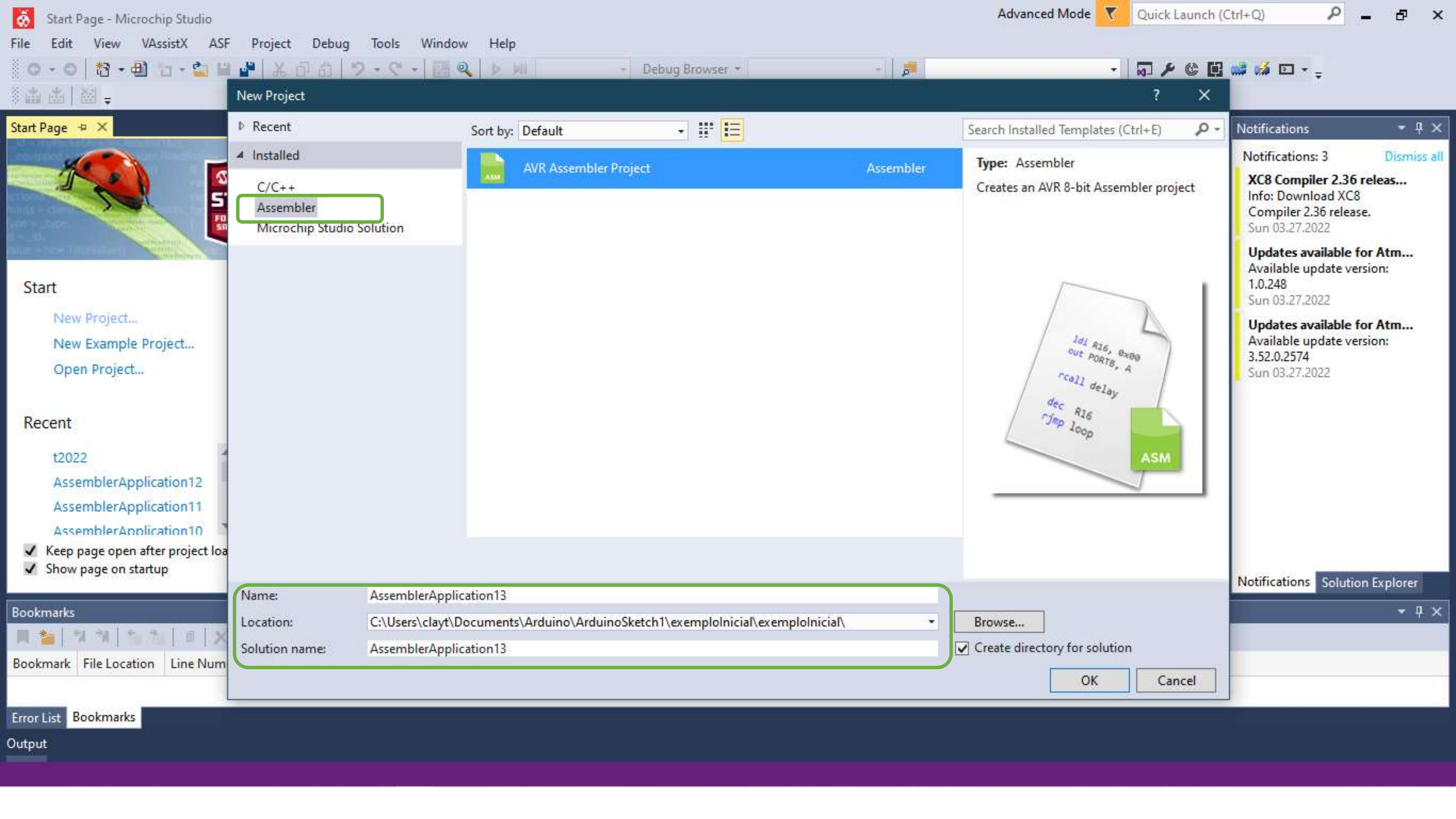
Atmel Studio is an Integrated Development Environment (IDE) for writing and debugging AVR[®]/ARM[®] applications in Windows[®] XP/Windows Vista[®]/Windows 7/8 environments. Atmel Studio provides a project management tool, source file editor, simulator, assembler, and front-end for C/C++, programming, and on-chip debugging.

Atmel Studio supports the complete range of Microchip AVR tools. Each new release contains the latest updates for the tools as well as support for new AVR/ARM devices.

Atmel Studio has a modular architecture, which allows interaction with 3rd party software vendors. GUI plugins and other modules can be written and hooked to the system. Contact [Microchip](#) for more information.

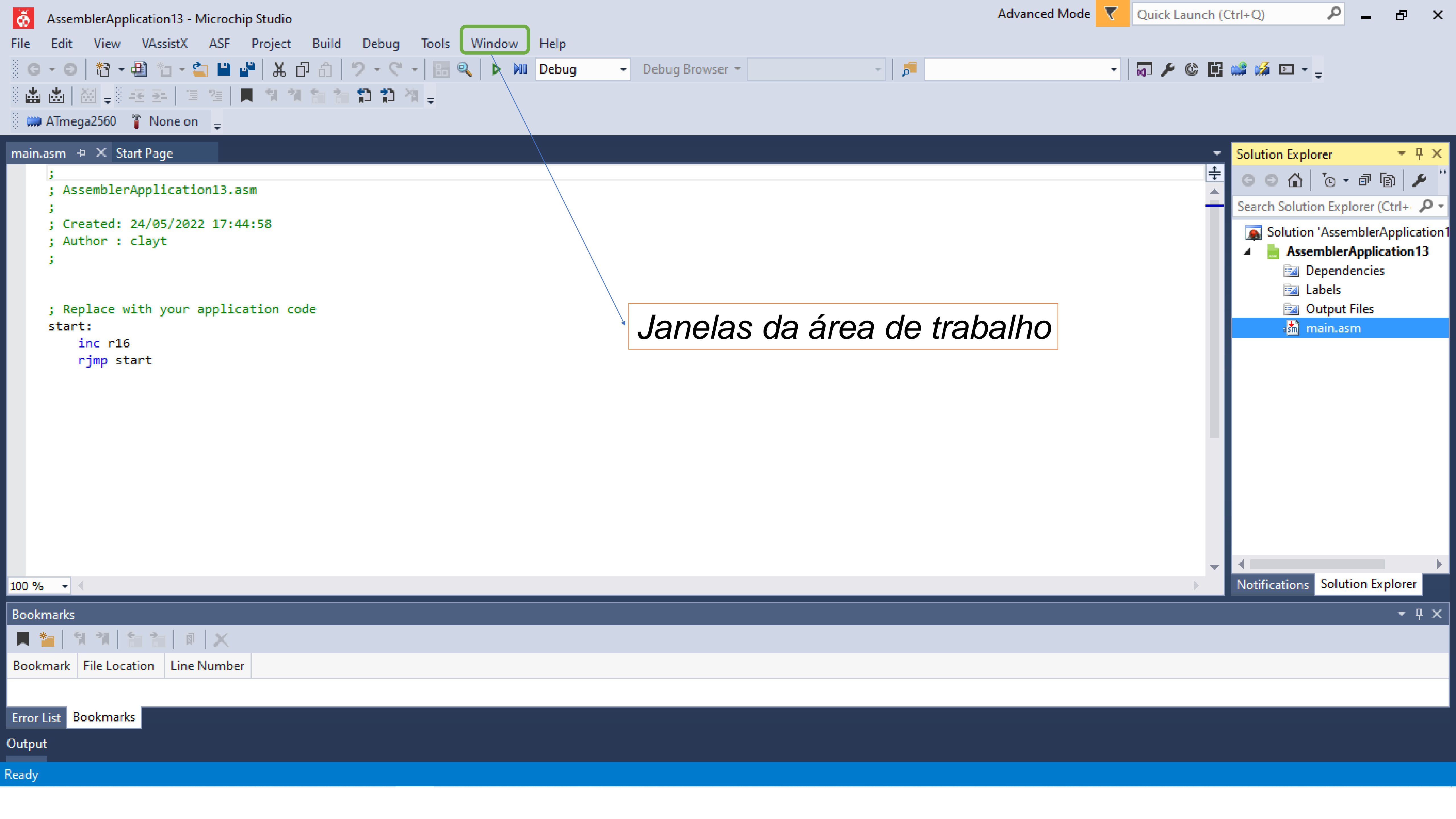




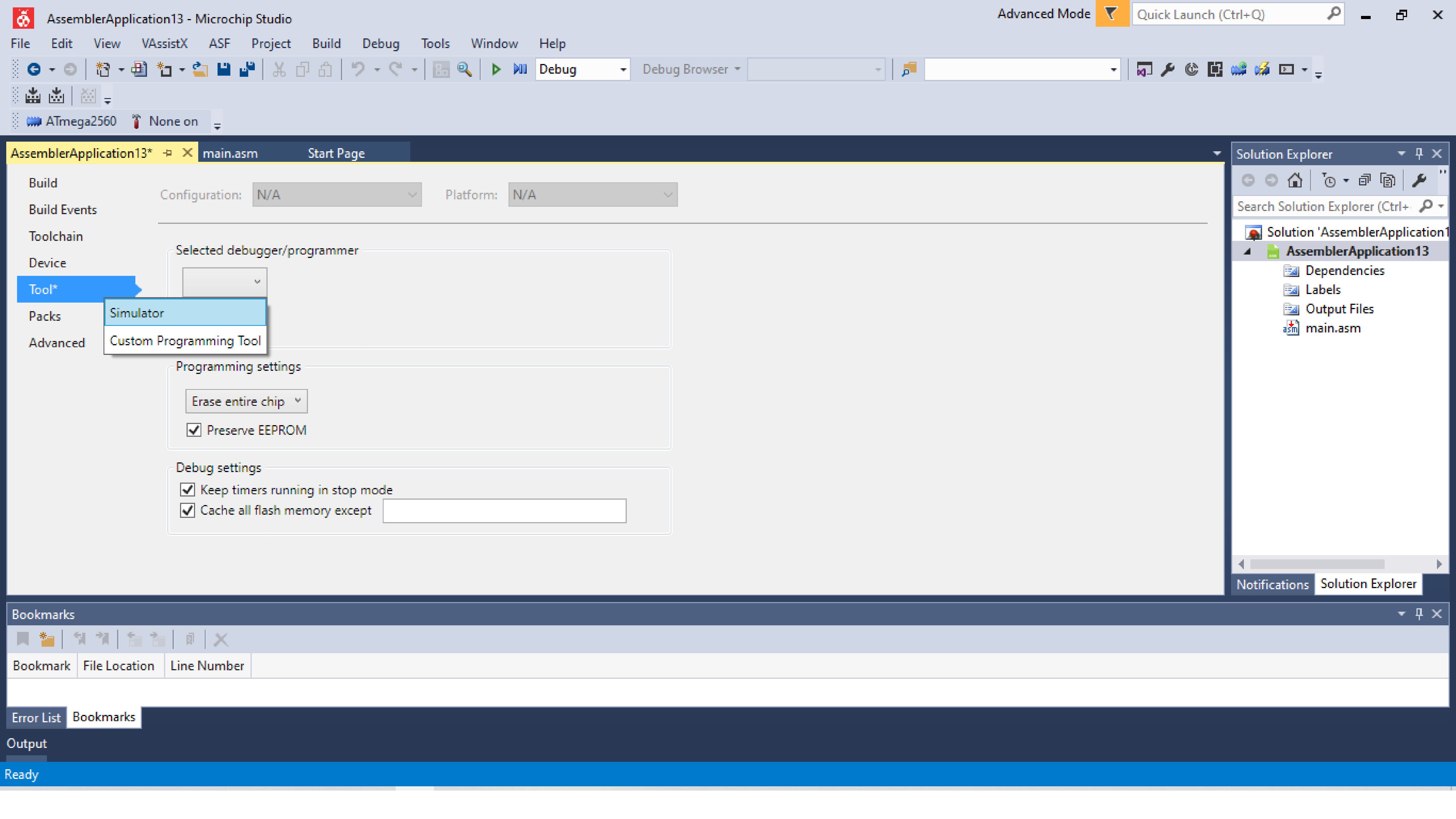


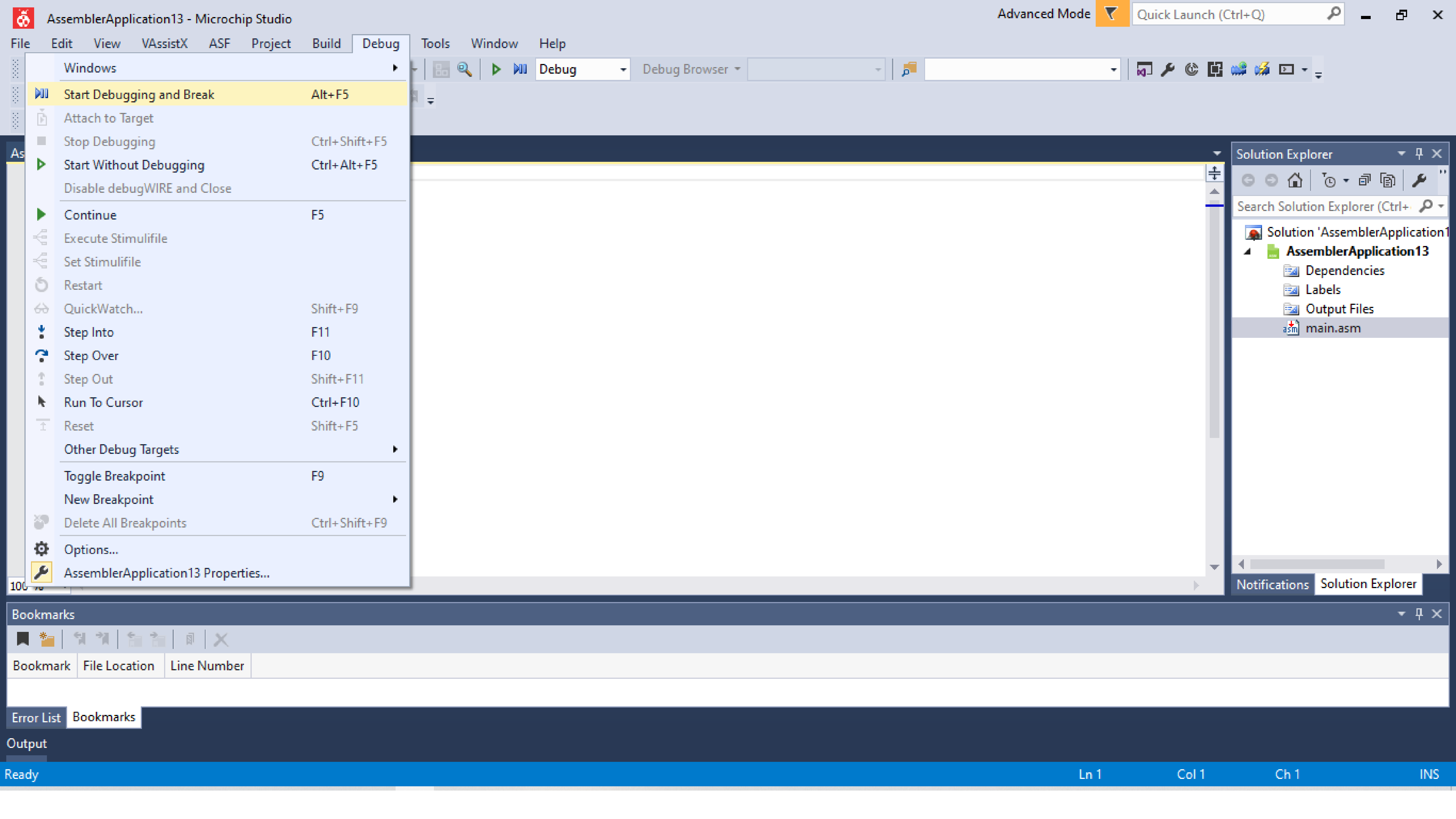
The screenshot shows the Microchip Studio IDE with the 'Device Selection' dialog open. The dialog lists various AVR microcontrollers. The 'ATmega2560' is highlighted in blue. To the right, the 'Device Info' panel provides specific details for the selected device. The background shows the IDE's start page with project creation options and a sidebar with 'Notifications' and 'Solution Explorer' tabs.

Name	App./Boot Memory (Kbytes)	Data Memory (bytes)	EEPROM (bytes)
ATmega169A	16	1024	512
ATmega169P	16	1024	512
ATmega169PA	16	1024	512
ATmega16A	16	1024	512
ATmega16HVA	16	512	256
ATmega16HVB	16	1024	512
ATmega16HVBrevB	16	1024	512
ATmega16M1	16	1024	512
ATmega16U2	16	512	512
ATmega16U4	16	1280	512
ATmega2560	256	8192	4096
ATmega2561	256	8192	4096
ATmega2564RFR2	256	32768	8192
ATmega256RFR2	256	32768	8192
ATmega32	32	2048	1024
ATmega3208	32	4096	256
ATmega3209	32	4096	256
ATmega324A	32	2048	1024
ATmega324P	32	2048	1024



Janelas da área de trabalho





Windows

Start Debugging and BreakAlt+F5

Attach to Target

Stop DebuggingCtrl+Shift+F5

Start Without DebuggingCtrl+Alt+F5

Disable debugWIRE and Close

ContinueF5

Execute Stimulifile

Set Stimulifile

Restart

QuickWatch...Shift+F9

Step IntoF11

Step OverF10

Step OutShift+F11

Run To CursorCtrl+F10

ResetShift+F5

Other Debug Targets

Toggle BreakpointF9

New Breakpoint

Delete All BreakpointsCtrl+Shift+F9

Options...

AssemblerApplication13 Properties...

Debug Debug Browser

Solution Explorer

Search Solution Explorer (Ctrl+)

Solution 'AssemblerApplication13'

AssemblerApplication13

Dependencies

Labels

Output Files

main.asm

Notifications Solution Explorer

Bookmarks

Bookmark	File Location	Line Number

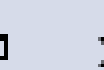
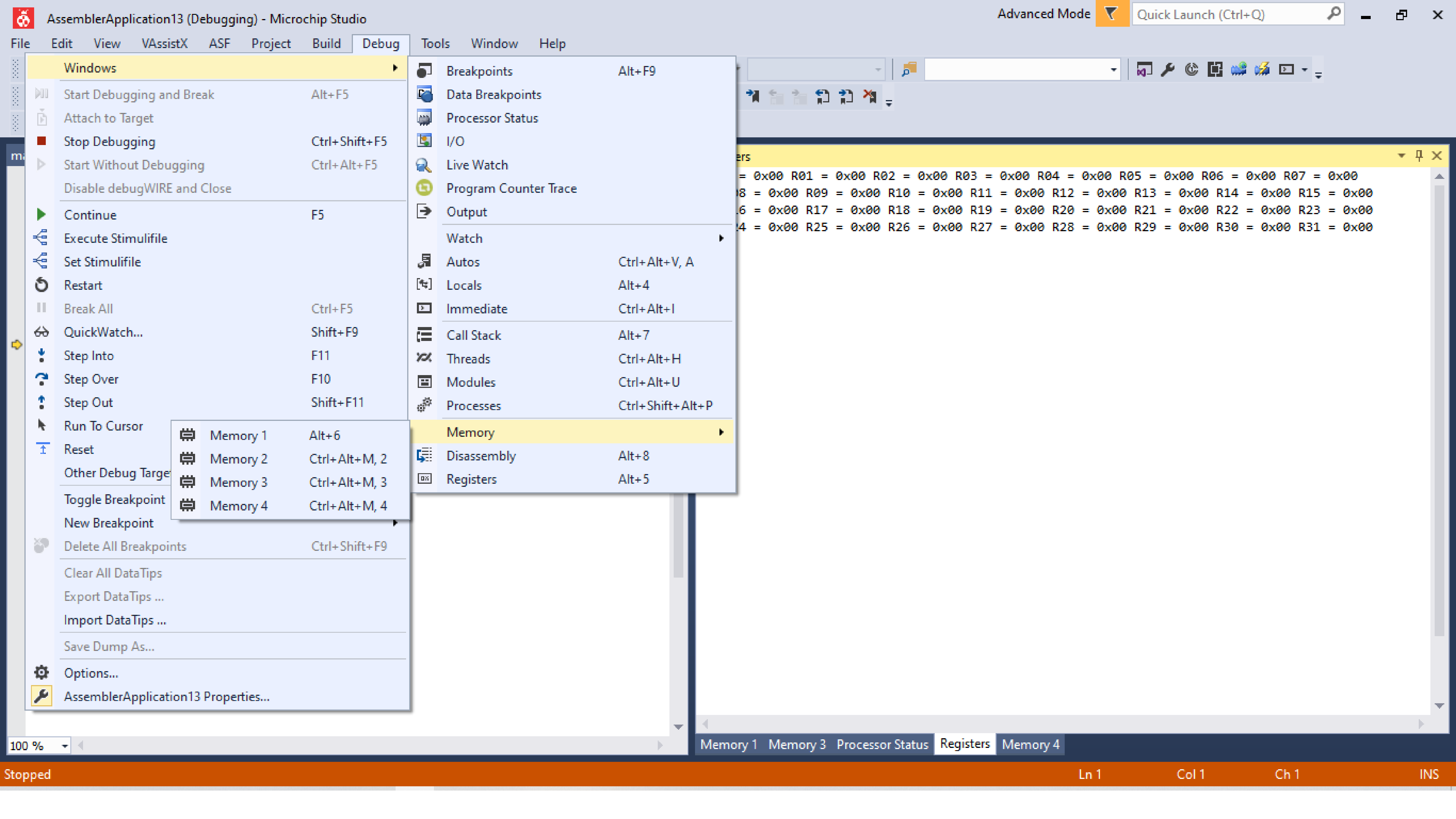

```

;
; AssemblerApplication13.asm
;
; Created: 24/05/2022 17:44:58
; Author : clayt
;

; Replace with your application code
start:
    inc r16
    rjmp start

```

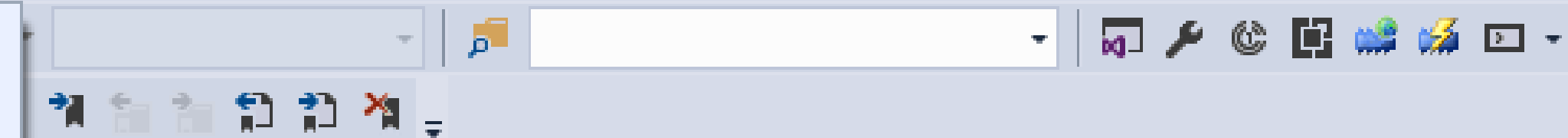
```
R00 = 0x00 R01 = 0x00 R02 = 0x00 R03 = 0x00 R04 = 0x00 R05 = 0x00 R06 = 0x00 R07 = 0x00
R08 = 0x00 R09 = 0x00 R10 = 0x00 R11 = 0x00 R12 = 0x00 R13 = 0x00 R14 = 0x00 R15 = 0x00
R16 = 0x00 R17 = 0x00 R18 = 0x00 R19 = 0x00 R20 = 0x00 R21 = 0x00 R22 = 0x00 R23 = 0x00
R24 = 0x00 R25 = 0x00 R26 = 0x00 R27 = 0x00 R28 = 0x00 R29 = 0x00 R30 = 0x00 R31 = 0x00
```



Windows

- Start Debugging and Break Alt+F5
- Attach to Target
- Stop Debugging Ctrl+Shift+F5
- Start Without Debugging Ctrl+Alt+F5
- Disable debugWIRE and Close
- Continue F5
- Execute Stimulifile
- Set Stimulifile
- Restart
- Break All Ctrl+F5
- QuickWatch... Shift+F9
- Step Into F11
- Step Over F10
- Step Out Shift+F11
- Run To Cursor
- Reset
- Other Debug Target
 - Memory 1 Alt+6
 - Memory 2 Ctrl+Alt+M, 2
 - Memory 3 Ctrl+Alt+M, 3
 - Memory 4 Ctrl+Alt+M, 4
- Toggle Breakpoint
- New Breakpoint
- Delete All Breakpoints Ctrl+Shift+F9
- Clear All DataTips
- Export DataTips ...
- Import DataTips ...
- Save Dump As...
- Options...
- AssemblerApplication13 Properties...

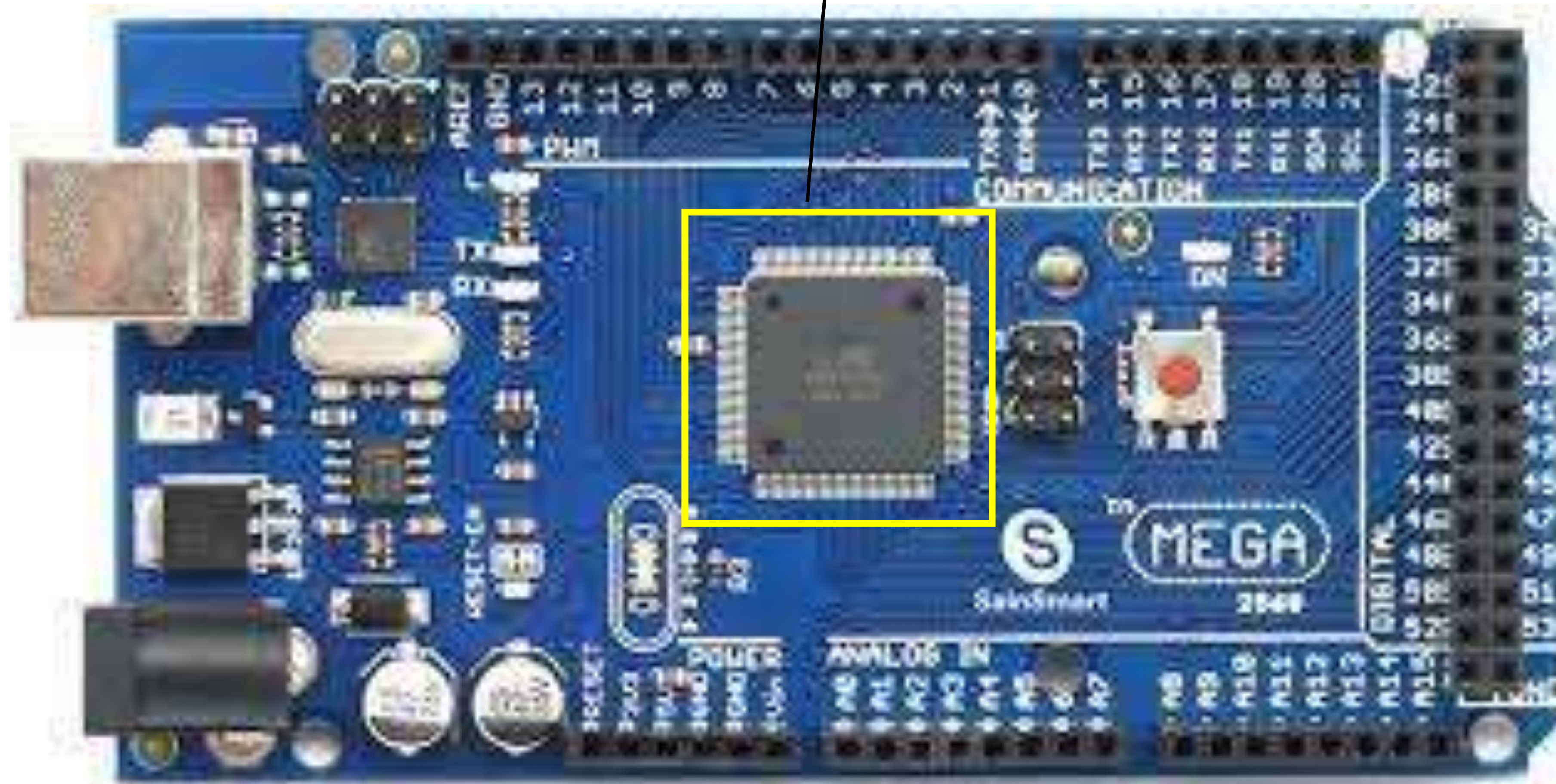
- Breakpoints Alt+F9
- Data Breakpoints
- Processor Status
- I/O
- Live Watch
- Program Counter Trace
- Output
- Watch
- Autos Ctrl+Alt+V, A
- Locals Alt+4
- Immediate Ctrl+Alt+I
- Call Stack Alt+7
- Threads Ctrl+Alt+H
- Modules Ctrl+Alt+U
- Processes Ctrl+Shift+Alt+P
- Memory
- Disassembly Alt+8
- Registers Alt+5



Registers

R0 = 0x00 R01 = 0x00 R02 = 0x00 R03 = 0x00 R04 = 0x00 R05 = 0x00 R06 = 0x00 R07 = 0x00
R08 = 0x00 R09 = 0x00 R10 = 0x00 R11 = 0x00 R12 = 0x00 R13 = 0x00 R14 = 0x00 R15 = 0x00
R16 = 0x00 R17 = 0x00 R18 = 0x00 R19 = 0x00 R20 = 0x00 R21 = 0x00 R22 = 0x00 R23 = 0x00
R24 = 0x00 R25 = 0x00 R26 = 0x00 R27 = 0x00 R28 = 0x00 R29 = 0x00 R30 = 0x00 R31 = 0x00

ATmega 2560



Start Page - Microchip Studio

Advanced ModeQuick Launch (Ctrl+Q)

FileEditViewVAssistXASFProjectDebugToolsWindowHelp

Start Page

Start

New Project...

New Example Project...

Open Project...

Recent

AssemblerApplication13t2022

AssemblerApplication12

AssemblerApplication11

Keep page open after project load

Show page on startup

Bookmarks

BookmarkFile LocationLine Num

Error ListBookmarks

Output

New Project

Recent

Installed

C/C++

Assembler

Microchip Studio Solution

Sort by: Default

AVR XC8 C Library ProjectC/C++

Convert AVR GCC to XC8 ProjectC/C++

GCC C ASF Board ProjectC/C++

GCC C Executable ProjectC/C++

GCC C Static Library ProjectC/C++

GCC C++ Executable ProjectC/C++

GCC C++ Static Library ProjectC/C++

SAM L11 Secure Solution v1.2C/C++

Create project from Arduino sketchC/C++

Search Installed Templates (Ctrl+E)

Type: C/C++

Creates an AVR 8-bit XC8 C application project

MPLAB XC COMPILER

Name: XC8Application1

Location: C:\Users\clayt\Documents\Arduino\ArduinoSketch1\exemplolnicial\exemplolnicial\

Solution name: XC8Application1

Browse...

Create directory for solution

OKCancel

Solution Explorer

NotificationsSolution Explorer

Aplicações em *assembly* para o AVR *Assembler*

Sintaxe

4.	AVR Assembler Syntax.....	12
4.1.	Keywords.....	12
4.2.	Preprocessor Directives.....	12
4.3.	Comments.....	12
4.4.	Line Continuation.....	12
4.5.	Integer Constants.....	12
4.6.	Strings and Character Constants.....	12
4.7.	Multiple Instructions per Line.....	13
4.8.	Operands.....	13

Instruções do *set* de instruções

O *AVR Assembler* reconhece instruções do *set* de instruções do processador diretamente convertidas para o processador.

Exemplos:

ADC – Add with Carry.....	23
ADD – Add without Carry.....	24
ADIW – Add Immediate to Word.....	25
AND – Logical AND.....	26
ANDI – Logical AND with Immediate.....	27
ASR – Arithmetic Shift Right.....	28
BCLR – Bit Clear in SREG.....	29
BLD – Bit Load from the T Bit in SREG to a Bit in Register.....	30
BRBC – Branch if Bit in SREG is Cleared.....	31
BRBS – Branch if Bit in SREG is Set.....	32
BRCC – Branch if Carry Cleared.....	33
BRCS – Branch if Carry Set.....	34
BREAK – Break.....	35
BREQ – Branch if Equal.....	35
BRGE – Branch if Greater or Equal (Signed).....	36
BRHC – Branch if Half Carry Flag is Cleared.....	37
BRHS – Branch if Half Carry Flag is Set.....	38
BRID – Branch if Global Interrupt is Disabled.....	39

Diretivas

O *assembler* reconhece também as chamadas **diretivas**, que não são os comandos do processador, mas orientam o comportamento do *assembler* na transformação do nível de montagem para o nível ISA.

Exemplos:

5. Assembler Directives.....	14
5.1. BYTE.....	14
5.2. CSEG.....	14
5.3. CSEGSIZE.....	14
5.4. DB.....	15
5.5. DD.....	15
5.6. DEF.....	16
5.7. DQ.....	16
5.8. DSEG.....	16
5.9. DW.....	17
5.10. ELIF and ELSE.....	17
5.11. ENDIF.....	17
5.12. ENDM and ENDMACRO.....	18
5.13. EQU.....	18
5.14. ERROR.....	18
5.15. ESEG.....	19
5.16. EXIT.....	19
5.17. IF, IFDEF, and IFNDEF.....	19
5.18. INCLUDE.....	20
5.19. LIST.....	20
5.20. LISTMAC.....	21
5.21. MACRO.....	21
5.22. MESSAGE.....	21

Label

As instruções de desvio podem remeter a um endereço ou a um *label* (rótulo), que marca a linha do código para a qual a próxima instrução deve ser executada.

Exemplo

```
; Exemplo label  
ldi r26,0x00  
ldi r27,0x02  
jmp rotulo  
ldi r19,0x11  
rotulo: ldi r19,0x33  
st x,r19
```

Macros

A diretiva MACRO informa ao Assembler que este é o início de uma macro. Leva o **nome da macro** como parâmetro e pode ter até 10 parâmetros. A definição da macro é encerrada por uma diretiva ENDMACRO.

Sintaxe:

```
;Realiza a soma de dois registradores r16 e r17 e carrega na posição de memória definida por X
.MACRO soma
add r16,r17
st X,r16
inc r26
jmp rotulo
.ENDMACRO

;Exemplo MACRO
ldi r27,0x02 ;Inicializa X
ldi r26,0x00
ldi r16,0x03
ldi r17,0x2a
rotulo: soma
```


Diretiva .def

Define um nome simbólico em um registrador. Permite que sejam referenciados por meio do símbolo e que pode ser usado no programa.

Sintaxe:

```
;Atribui representacao simbolica aos registradores
```

```
.def var1=r16
```

```
.def var2=r17
```

```
.def x1=r27
```

```
.def x0=r26
```

```
;Define MARCO
```

```
.MACRO soma
```

```
add var1,var2
```

```
st X,var1
```

```
inc x0
```

```
jmp rotulo
```

```
.ENDMACRO
```

```
;Corpo do codigo
```

```
ldi x1,0x02
```

```
ldi x0,0x00
```

```
ldi var1,0x03
```

```
ldi var2,0x2a
```

```
rotulo:soma
```

Diretivas *.equ/.set*

- Definem um símbolo igual a uma expressão. Atribuem um valor a um rótulo, que ser usado em expressões posteriores.
- *.EQU*. O rótulo é uma constante e não pode ser alterado ou redefinido.
- *.SET*. Ao contrário do EQU, o rótulo pode ser alterado (redefinido) posteriormente no programa.

Sintaxe:

```
;Atribui representacao simbolica aos registradores
.def var1=r16
.def var2=r17
.equ delta=0x030a

;Define MARCO
.MACRO soma
add var1,var2
st X,var1
inc r26
jmp rotulo
.ENDMACRO

;Corpo do codigo
ldi r27,0x02
ldi r26,0x00
ldi var1,0x03
lds var2,delta
rotulo:soma
```

```
;Atribui representacao simbolica aos registradores
.def var1=r16
.def var2=r17
.set delta=0x030a

;Define MARCO
.MACRO soma
add var1,var2
st X,var1
inc r26
.set delta=delta+0x0002
jmp rotulo
.ENDMACRO

;Corpo do codigo
ldi r27,0x02
ldi r26,0x00
ldi var1,0x03
lds var2,delta
rotulo:soma
```


Diretivas .cseg/.dseg

- *.CSEG*. Segmento de código. Define o início de um Segmento de Código. Um arquivo pode ter vários segmentos distintos, que são concatenados quando montados. A diretiva não leva nenhum parâmetro.
- *.DSEG*. Segmento de dados. Define o início de um segmento de dados. Um arquivo pode consistir em vários segmentos de dados, que são concatenados em um único segmento de dados quando montados. A diretiva não leva nenhum parâmetro

Diretivas .cseg/.dseg

```
;Atribui representacao simbolica aos registradores
```

```
.DSEG
```

```
.def var1=r16
```

```
.def var2=r17
```

```
.set delta=0x030a
```

```
.CSEG
```

```
;Define MARCO
```

```
.MACRO soma
```

```
add var1,var2
```

```
st X,var1
```

```
inc r26
```

```
.set delta=delta+0x0002
```

```
jmp rotulo
```

```
.ENDMACRO
```

```
;Corpo do codigo
```

```
ldi r27,0x02
```

```
ldi r26,0x00
```

```
ldi var1,0x03
```

```
lds var2,delta
```

```
rotulo:soma
```


Aplicações - Exemplos

Exemplo 1

Carregar os valores decimais 1 a 16 nos registradores internos do processador R16 a R31.

Exemplo 1

- Instruções *ldi rd,K*
 onde $16 \leq d \leq 31$, $0 \leq K \leq 255$

[illegible]

Exemplo 2

Carregar os valores decimais 1 a 16 nos registradores internos do processador R16 a R31. Somar todos os valores.

Exemplo 3

Elaborar um código para carregar o dado 15 no endereço 0x0200, da SRAM do microcontrolador 2560. O programa deve armazenar os 10 múltiplos de 3 a partir de 15 nos endereços seguintes ao endereço 0x0200.

Exemplo 3

Elaborar um código para carregar o dado 15 no endereço 0x0200, da SRAM do microcontrolador 2560. O programa deve armazenar os 10 múltiplos de 3 a partir de 15 nos endereços seguintes ao endereço 0x0200.

```
.DSEG
.equ ref1=0x02
.equ ref0=0x00 ;Endereco inicial
.set lim=0x0f ;Valor inicial a armazenar
.equ num=0x03 ;Valor de base
.equ flag=0x0b ;Numero de multiplos de 3 a armazenar

.CSEG
.MACRO soma
.set lim=lim+3
inc r26
jmp loop
.ENDMACRO

ldi r27,ref1
ldi r26,ref0
loop:ldi r16,lim
st x,r16
cpi r26,flag
brne bloco
break
bloco: soma
```


Exemplo 4

Elaborar um código para ler um dado de 8 bits de um dispositivo de entrada conectado no registrador 0 de I/O do microcontrolador. Carregar o dado lido no endereço 0x0200, da SRAM. O programa deve pesquisar se o dado está em uma tabela nos endereços 0x0201 a 0x020A. Se o dado for encontrado, o sistema deve apresentar no registrador 1 de I/O, para o dispositivo de saída, o dado armazenado no endereço 0x020B. Em caso contrário, o sistema deve apresentar na porta B, para o dispositivo de saída, o dado armazenado no endereço 0x020C.

Exemplo 4

Elaborar um código para ler um dado de 8 bits de um dispositivo de entrada conectado no registrador 0 de I/O do microcontrolador. Carregar o dado lido no endereço 0x0200, da SRAM. O programa deve pesquisar se o dado está em uma tabela nos endereços 0x0201 a 0x020A. Se o dado for encontrado, o sistema deve apresentar no registrador 1 de I/O, para o dispositivo de saída, o dado armazenado no endereço 0x020B. Em caso contrário, o sistema deve apresentar na porta B, para o dispositivo de saída, o dado armazenado no endereço 0x020C.

```
.DSEG
.equ ref1=0x02
.equ ref0=0x00
.equ entrada=0x00
.set lim=0x0201
.equ acerto=0x020B
.equ falta=0x020C
.equ flag=0x0b

.CSEG
in r16,entrada
ldi r27,ref1
ldi r26,ref0
st x,r16
loop: lds r17,lim
cp r16,r17
brne bloco
lds r18,acerto
out 1,r18
break
bloco: .set lim=lim+1
cpi r26,flag
brne loop
lds r18,falta
out 1,r18
break
```




IBMEC.BR

 /IBMEC

 IBMEC

 @IBMEC_OFICIAL

 @IBMEC

 **ibmec**