Introduction to TMS320F28335[*]

Frank Bormann

Abstract

A basic overview of the TMS320F28335 microcontroller, its capabilities, and its functions.

Welcome to the F2833x - Tutorial

Welcome to the Texas Instruments TMS320F28335 Tutorial. This material is intended to be used as a student guide for a series of lessons and lab exercises dedicated to the TMS320F28335 Digital Signal Controller. The series of modules will guide you through the various elements of this device and train you in using Texas Instruments development tools as well as additional Internet resources.

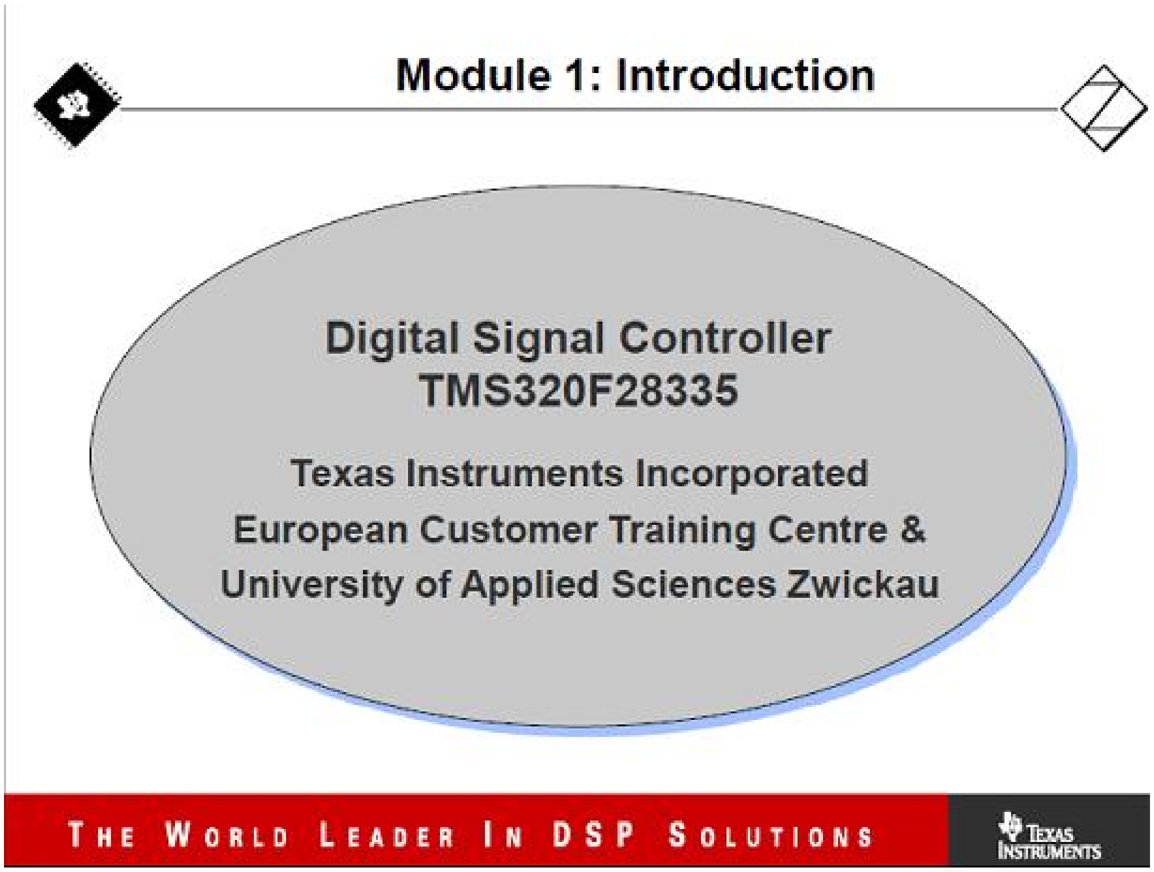
The material should be used for undergraduate classes at university. A basic knowledge of microprocessor architecture and programming microprocessors in language C is necessary. The material in Modules 1 to 10 shall be used in one semester, accompanied by lab exercises in parallel. Each module includes a detailed lab procedure for self study and guidance during the lab sessions.

The experimental lab sessions are based on the Texas Instruments "Peripheral Explorer Board" (TI part number: TMDSPREX28335). A 32K code-size limited version of the software design suite "Code Composer Studio" is bundled with the Peripheral Explorer Board and will be used for the development of code examples.

Modules 11 to 19 of the series go deeper into details of the TMS320F28335. They cover more advanced subjects and can be viewed as an optional series of lessons.

[*]Version 1.2: Jan 25, 2011 12:27 pm US/Central

[†]http://creativecommons.org/licenses/by/3.0/



Module 1: Introduction

Digital Signal Controller
TMS320F28335

Texas Instruments Incorporated
European Customer Training Centre &
University of Applied Sciences Zwickau

THE WORLD LEADER IN DSP SOLUTIONS

TEXAS INSTRUMENTS

Figure 1

Course Structure

Chapter 1: Introduction to Microprocessor, MCU and DSP
Chapter 2: TMS320F28335 Architecture
Chapter 3: Software Development Tools
Chapter 4: Fixed Point, Floating Point or both?
Chapter 5: Digital Input/Output
Chapter 6: Understanding the F28335 Interrupt System
Chapter 7: Control Actuators and Power Electronics
Chapter 8: Sensor Interface - Analogue to Digital Converter
Chapter 9: Communication I: Serial Communication Interface
Chapter 10: Communication II: Serial Peripheral Interface
Chapter 11: Communication III: Controller Area Network (CAN)
Chapter 12: Communication IV: Inter Integrated Circuit®
Chapter 13: Communication V: Multi Channel Buffered SerialPort
Chapter 14: Internal FLASH Memory and stand alone control
Chapter 15: Boot – loader and Field update

Chapter 16: FLASH – Application Program Interface (API)
Chapter 17: IQ-Math and floating point hardware
Chapter 18: Digital Motor Control
Chapter 19: Digital Power Supply

Installation and Laboratory Preparation

This paragraph is for teachers / instructors only. If you read this textbook as a student, please continue at Page 1-8.

The following preparations are necessary to use and run the laboratory exercises of this tutorial:

1. A valid and working version of Code Composer Studio, Version 4.1 should be installed. The default folder is "C:\Program Files\Texas Instruments\ccsv4". Some of the examples will refer to the subfolder "C:\Program Files\Texas Instruments\ccsv4\C2000" to access C run-time support libraries and linker command files. All laboratory procedures will reference this location. Please inform your students if your installation was made to a different directory.

If you do not have a valid CCS4.1 installation, please use the CCS4.1 - DVD, which comes with the Peripheral Explorer Board. To install and setup CCS4.1, please follow the guidelines from the Texas Instruments wiki-page (http://processors.wiki.ti.com/index.php/C2000_Getting_Started_with_Code_Composer_Studio_v4).

2. The Peripheral Explorer Board "Quick Start Guide" (sprugm2.pdf) describes the hardware environment in detail. The file "sprugm2.pdf" can be found in folder "C2833x_CCS4/hardware" of the C2000 teaching CD-ROM.

3. Install the Peripheral Register Header File support package. All laboratory exercises expect to find the Header File package version 1.31 (sprc530.zip) installed in folder "C:\tidcs\c28\DSP2833x\v131". If it is not yet installed, you can find the zip-file at the teaching CD-ROM under "C2833x_CCS4/libraries".

4. For IQ-Math based exercises you must have the IQ-Math library (sprc087.zip). The default location is "C:\tidcs\c28\IQmath\v15a". If this library is not yet installed, you can find the zip-file at the teaching CD-ROM under "C2833x_CCS4/libraries".

Piccolo F28027-USB stick

This teaching CD-ROM and all Laboratory exercises are based on the TMS320F28335 processor and the Peripheral Explorer Board. For the TMS320F28027 Texas Instruments offers an ultra low cost evaluation kit (TMDX28027USB):

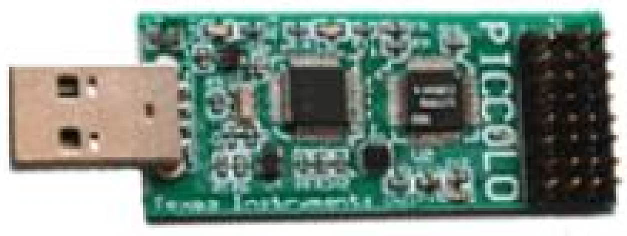


Figure 2

The teaching CD-ROM dedicates an entire chapter (Module 7A) for the most important peripheral unit, the Pulse Width Modulation Unit (PWM), which describes the laboratory procedures and provides the laboratory templates and solutions for more than 10 exercises. These documents can be found in folder "C2833x_CCS4/Modules/Module7" of the teaching CD-ROM (files "Module_07_A.pdf"and "Solution_07_A.zip").

For peripheral units of the F28027, you can simply modify the source code of the F28335 examples on this CD-ROM and run the examples with a F28027. Of course, the USB-stick does not have all the additional external devices of the Peripheral Explorer Board, which must be added manually to the USB stick. For convenience, the corresponding Header File support package (sprc832.zip) for the F28027 is also part of the teaching CD-ROM, as well as some additional support files (sprc835.zip). Both files can be found in folder "C2833x_CCS3/libraries" of the teaching CD-ROM.

Template Files for Laboratory Exercises

All modules after module two are accompanied by laboratory exercises. For some of the modules, template files are provided with the CD ("lab template files"), for other modules, the students are expected to develop their own project files out of previous laboratory sessions. In these cases, the lab description in the textbook chapter explains the procedure. A 2nd group of project files ("solution files") provides a full solution directory for all laboratory exercises. This group is intended to be used by teachers only. Instead of a single zip-file for the whole CD-ROM we decided to use separate archive files for the individual modules. This gives the teacher the opportunity to select parts of the CD to be used in his classes.

The zip-files may be extracted to a working directory of your choice. **However, the textbook assumes that the files are located in: "C:\DSP2833x\Labs" for group #1 and "C:\DSP2833x\solution" for group #2.** When extracted, a subfolder named as the exercise number will be added.

The laboratory exercises are:

Lab3: "Beginner's project" - basic features of Code Composer Studio

Lab4_1: "Numbering Systems" - fixed-point multiply operation

Lab4_2: "Numbering Systems" - floating-point multiply (hardware and software)

Lab5_1: "Digital Output" - 4 LEDs binary counter-sequence

Lab5_2: "Digital Output" - 4 LEDs blinking "knight-rider"

Lab5_3: "Digital Input" - read 4 bit hexadecimal encoder and display value

Lab5_4: "Digital Input / Output" - speed control of binary counter by hex-encoder

Lab5_5: "Digital Input / Output" - additional start/stop push-buttons

Lab6: "CPU-Timer 0 and Interrupts" - add a hardware timer to Lab5_1 and use an interrupt service routine (hardware time base framework)

Lab7_1: "Pulse Width Modulation" - generate a single ePWM (e = "enhanced") output signal

Lab7_2: "3 – Phase PWM" - generate a phase shifted set of 3 ePWM – signals

Lab7_3: "variable Pulse Width" - generate a 1 kHz – signal with variable pulse width

Lab7_4: "dual complementary PWM" - generate a pair of complementary PWM signals

Lab7_5: "dual channel modulation" - independent modulation of pulse width at ePWMA and ePWMB

Lab7_6: "Dead Band Generator" - generate a dead band delay at ePWMA and ePWMB

Lab7_7: "Chopper Mode Unit" - split the active pulse phases in a series of high frequency pulses

Lab7_8: "Trip Zone Protection" - switch off power lines in case of an over – current

Lab7_9: "Sinusoidal Signal" - use ePWM to generate sinusoidal signals (class D audio amplifiers)

Lab7_10:"Capture Unit" - use a capture unit to measure a 1 kHz - signal

Lab7_11:"Radio Remote Control Unit" - use a capture unit to receive and decode an infrared radio remote control unit (RC5-code)

Lab8_1: "ADC dual conversion" - convert two analogue input voltages

Lab8_2: "ADC and control" - speed control of binary counter by ADCINA0

Lab9_1: "SCI - transmission" - send text message "F28335 UART is fine!"

Lab9_2: "SCI – transmit interrupts" - use of SCI – transmit interrupt services

Lab9_3: "SCI – transmit FIFO" - use of SCI – FIFO for transmission

Lab9_4: "SCI – receive and transmit" - wait for message "Texas" and answer with "Instruments"

Lab9_5: "SCI – remote control" - control speed of binary counter by SCI – message

Lab11_1:"CAN – Transmission" - periodic transmission of a binary counter at 100 kbit/s and Identifier 0x1000 0000

Lab11_2: "CAN - Reception" - Receive Identifier 0x1000 0000 at 100 kbit/s and display the message at 2 LED's.

Lab11_3:“CAN – Transmit & Receive” - merger of Lab11_1 and Lab11_2
Lab11_4:”CAN – Interrupt” - use of CAN – interrupts to receive messages
Lab11_5:”CAN – Error – Handling” - use of CAN – error interrupts
Lab11_6:”CAN – Remote Transmit Request” – use of CAN transmit requests
Lab12_1:”I2C – Temperature Sensor” - use of TMP101 in 9 bit resolution mode
Lab12_2:”I2C – Temperature Sensor” - use of TMP101 in 12 bit resolution mode
Lab12_3:”I2C – Temperature Sensor” - use of I2C –FIFO – registers for TMP101
Lab12_4:”I2C – Temperature Sensor” - use of I2C – Interrupt System
Lab13_1:”McBSP and SPI” - use of audio codec AIC23B to generate a single sinusoidal tone
Lab13_2:”McBSP and SPI” - use of audio codec AIC23B to generate a stereo sinusoidal tone
Lab13_3:”McBSP - Interrupts” - Lab13_2 plus McBSP – interrupt system
Lab13_4:”McBSP – SPI – Emulation” - Write and Read to an SPI – EEPROM AT25256
Lab 14_1: “Standalone FLASH” - change Lab6 to run directly from FLASH after power ON.
Lab 15_1: “SCI - Boot loader - download control code before start
Lab16_1: “FLASH – API” - update FLASH while the control code is running
Lab17: “IQ-MATH” - use of a digital low-pass filter in IQ-Math, generate a 2 KHz square wave signal, sample the signal with the Analogue to Digital Converter and calculate the low-pass filter.

Lab18: “Digital Motor Control” - Labs are based on Texas Instruments “Digital Motor Control Kit” (part number: TMDS2MTRPFCKIT); see laboratory descriptions, which are included in the software part of this kit.

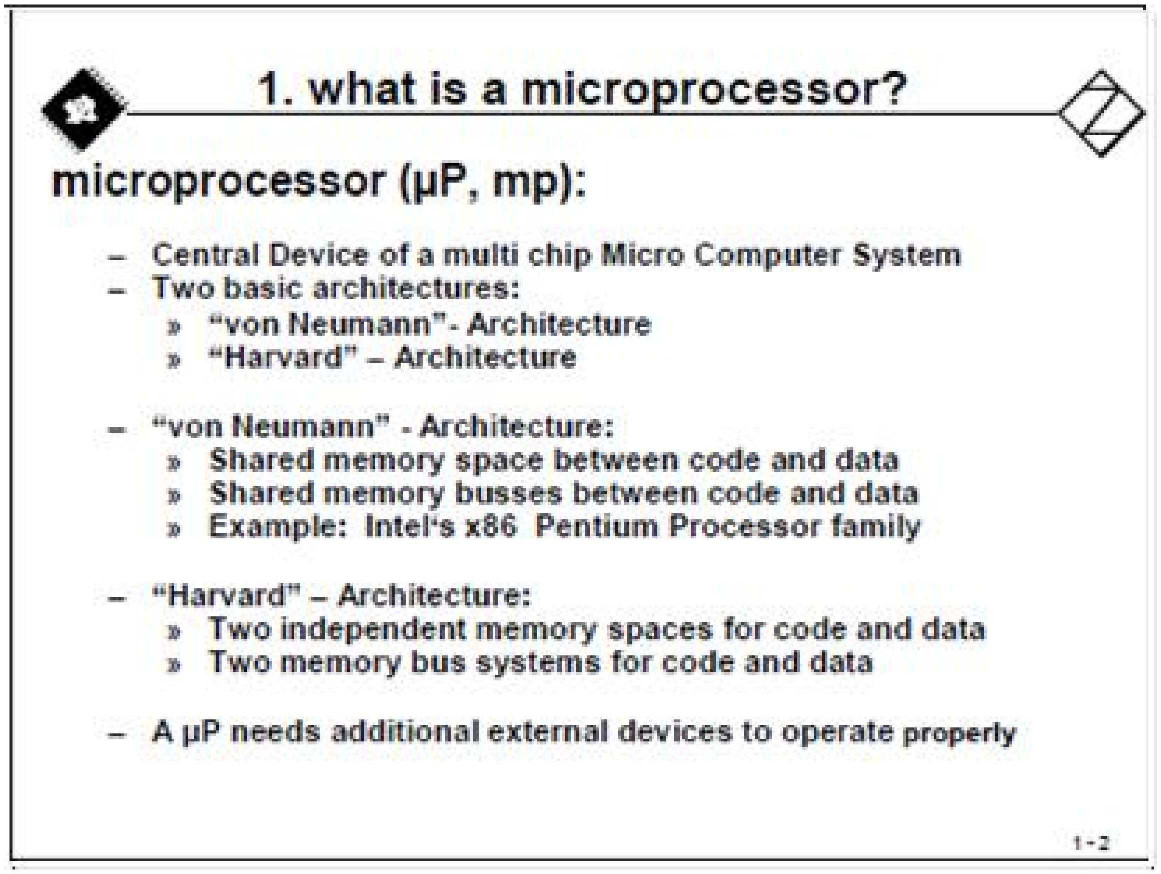
Lab19: “Digital Power Supply” - Labs are based on Texas Instruments “Digital Power Experimenter's Kit” (part number “TMDSDCDC2KIT”); see laboratory descriptions, which are included in the software part of this kit.

What is a Digital Signal Controller?

First we have to discus some keywords that are quite often used when we speak about digital control or computing in general. The TMS320F28335 belongs to a group of devices that are called “Digital Signal Controllers” (DSC). In computing, we use words like “Microprocessor”, “Microcomputer” or “Microcontroller” to specify a given sort of electronic device. When it comes to digital signal processing, the preferred name is “Digital Signal Processor” (DSP).

To begin with, let us introduce some terms:

- Microprocessor (μP)
- Micro Computer
- Microcontroller (μC)
- Digital Signal Processor (DSP)
- Digital Signal Controller (DSC)



1. what is a microprocessor?

microprocessor (μP, mp):

- Central Device of a multi chip Micro Computer System
- Two basic architectures:
  - » "von Neumann"- Architecture
  - » "Harvard" – Architecture
- "von Neumann" - Architecture:
  - » Shared memory space between code and data
  - » Shared memory busses between code and data
  - » Example: Intel's x86 Pentium Processor family
- "Harvard" – Architecture:
  - » Two independent memory spaces for code and data
  - » Two memory bus systems for code and data
- A μP needs additional external devices to operate properly

1-2

Figure 3

Microprocessors are based on a simple sequential procedural approach: 1) read next machine code instruction from code memory, 2) decode instruction, 3) read optional operands from data memory, 4) execute instruction, and 5) write back result. This series of events runs in an endless manner. To use a μP one has to add memory and additional external devices to the Microprocessor.

A typical microprocessor block diagram

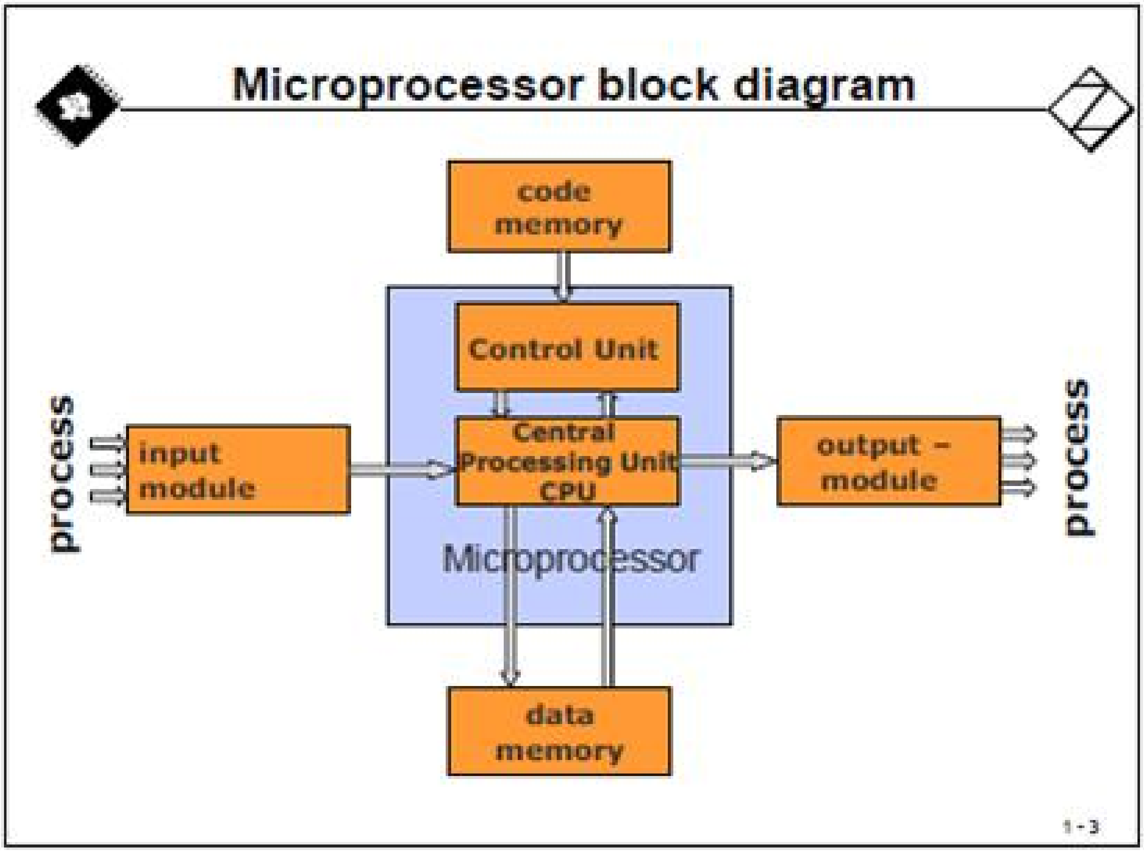


Figure 4

A typical microprocessor block diagram is shown above. As can be seen from Figure 4, the microprocessor consists of two parts – the control unit and the central processing unit (CPU). It operates on input signals, reads operands from data memory, writes results back in data memory, and updates output modules. All computing is based on machine code instructions, which are sequentially stored in code memory. The microprocessor reads these instructions one after another into its control logic.

The execution flow of a piece of machine code instructions follows a certain sequence, shown in the following slide. The life of a microprocessor is quite boring; it never goes off the beaten track unless it loses its power supply. The sequence is always:

1. Address the next entry in code memory
2. Read (or "fetch") the next machine instruction from this address
3. "Decode" that instruction and prepare next activities
4. Select one of five next steps:

- Read an input and compute it
- Read an entry from data memory and compute it
- Do an internal operation, which does not require an information exchange
- Write a result back in data memory
- Update an output channel with a result of a previous computation.

Note: Some processors are able to perform more than 1 step in parallel.
5. Calculate the next code memory address and return to step #1.

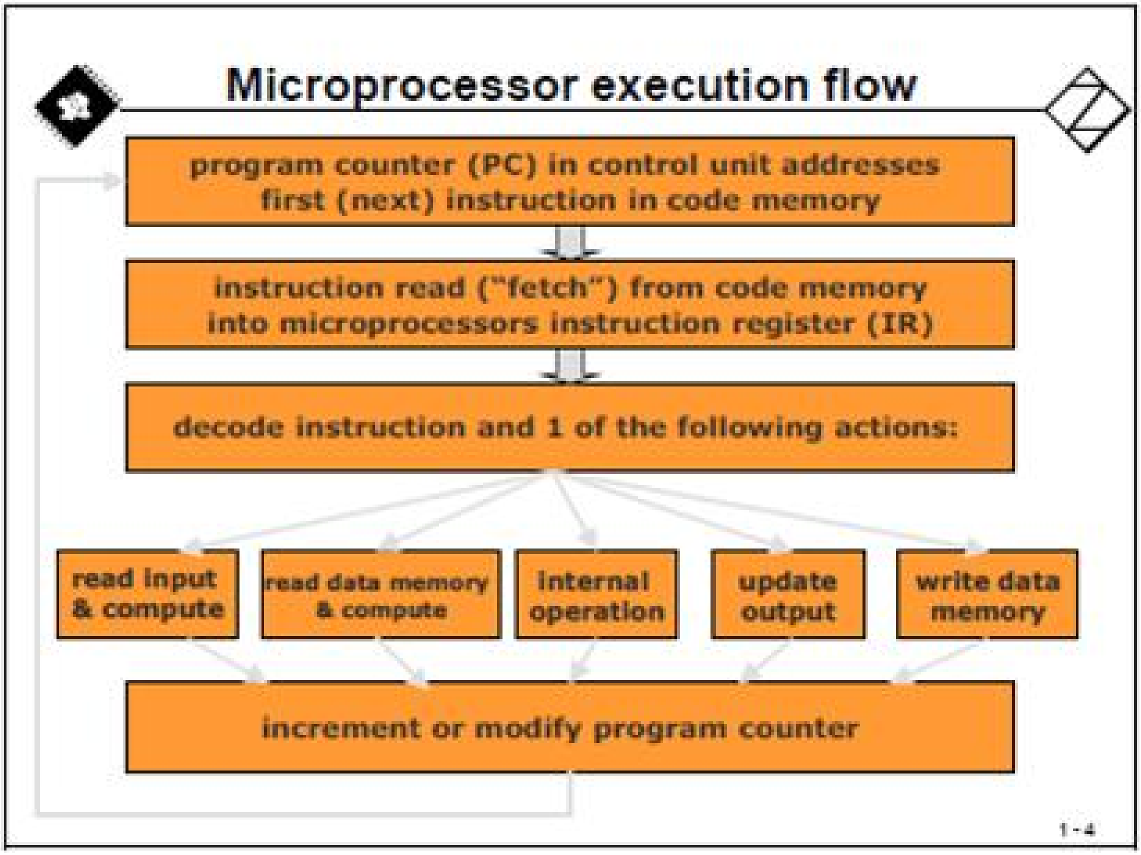
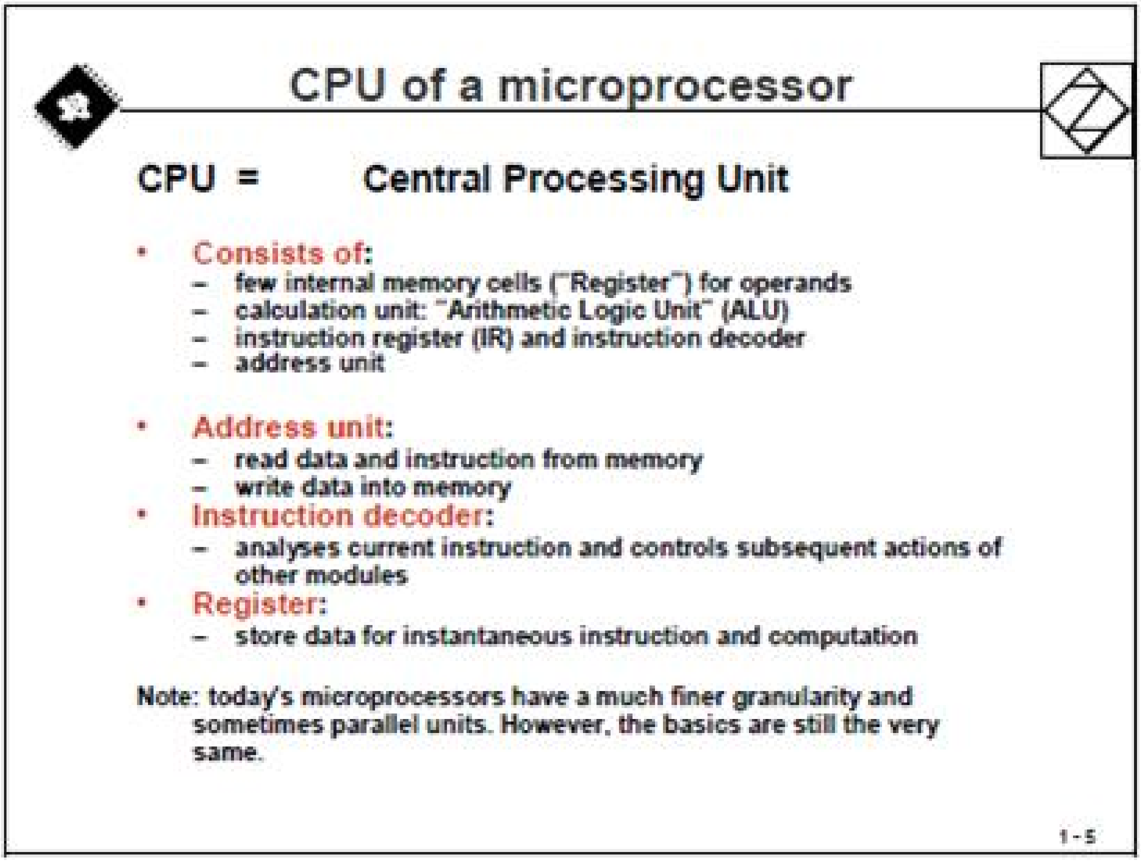

Microprocessor execution flow
program counter (PC) in control unit addresses first (next) instruction in code memory
instruction read ("fetch") from code memory into microprocessors instruction register (IR)
decode instruction and 1 of the following actions:
read input & compute
read data memory & compute
internal operation
update output
write data memory
increment or modify program counter
1-4

Figure 5

The heart of a microprocessor is its Central Processing Unit (CPU). To keep it simple, we just look at a very basic structure of a CPU. Today, a microprocessor is really one of the most complex integrated circuits.



CPU of a microprocessor

CPU = Central Processing Unit

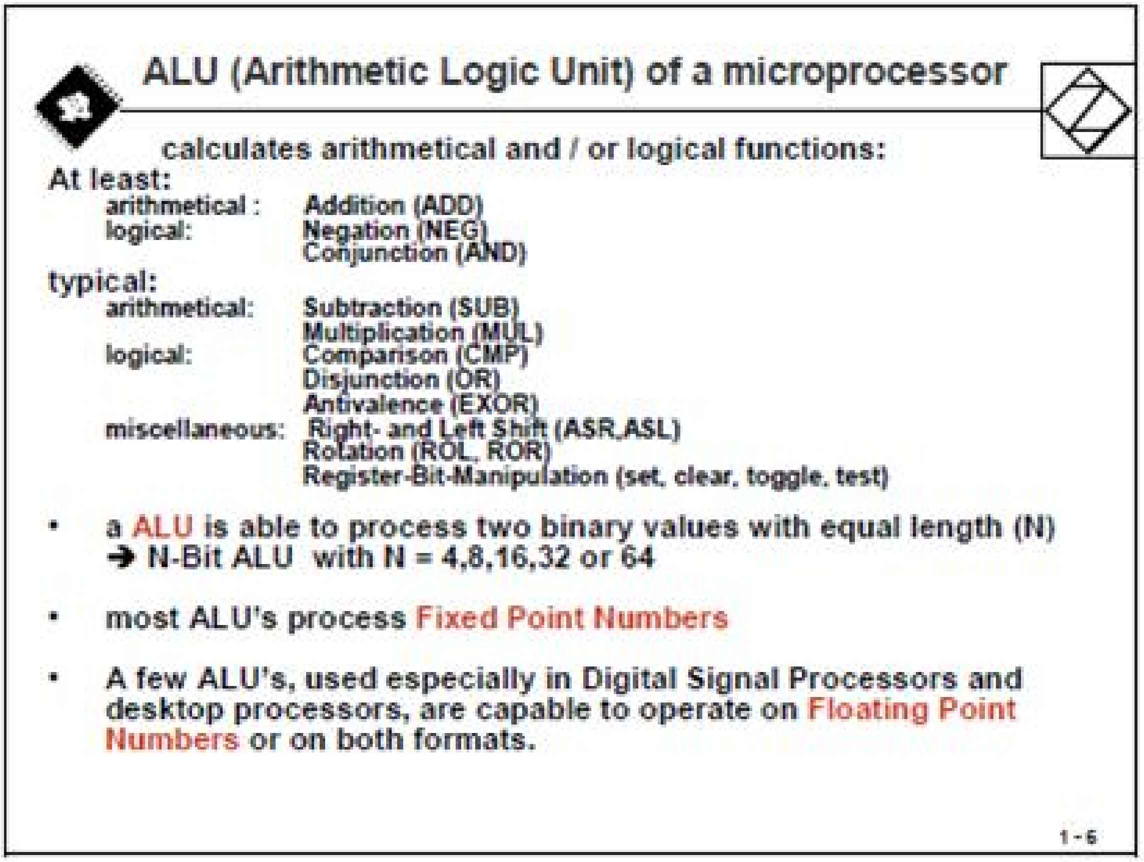
- Consists of:
  - few internal memory cells ("Register") for operands
  - calculation unit: "Arithmetic Logic Unit" (ALU)
  - instruction register (IR) and instruction decoder
  - address unit
- Address unit:
  - read data and instruction from memory
  - write data into memory
- Instruction decoder:
  - analyses current instruction and controls subsequent actions of other modules
- Register:
  - store data for instantaneous instruction and computation

Note: today's microprocessors have a much finer granularity and sometimes parallel units. However, the basics are still the very same.

1 - 5

Figure 6

Arithmethic Logic Unit ("ALU") of a microprocessor



ALU (Arithmetic Logic Unit) of a microprocessor

calculates arithmetical and / or logical functions:

At least:

arithmetical :	Addition (ADD)
logical:	Negation (NEG)
	Conjunction (AND)

typical:

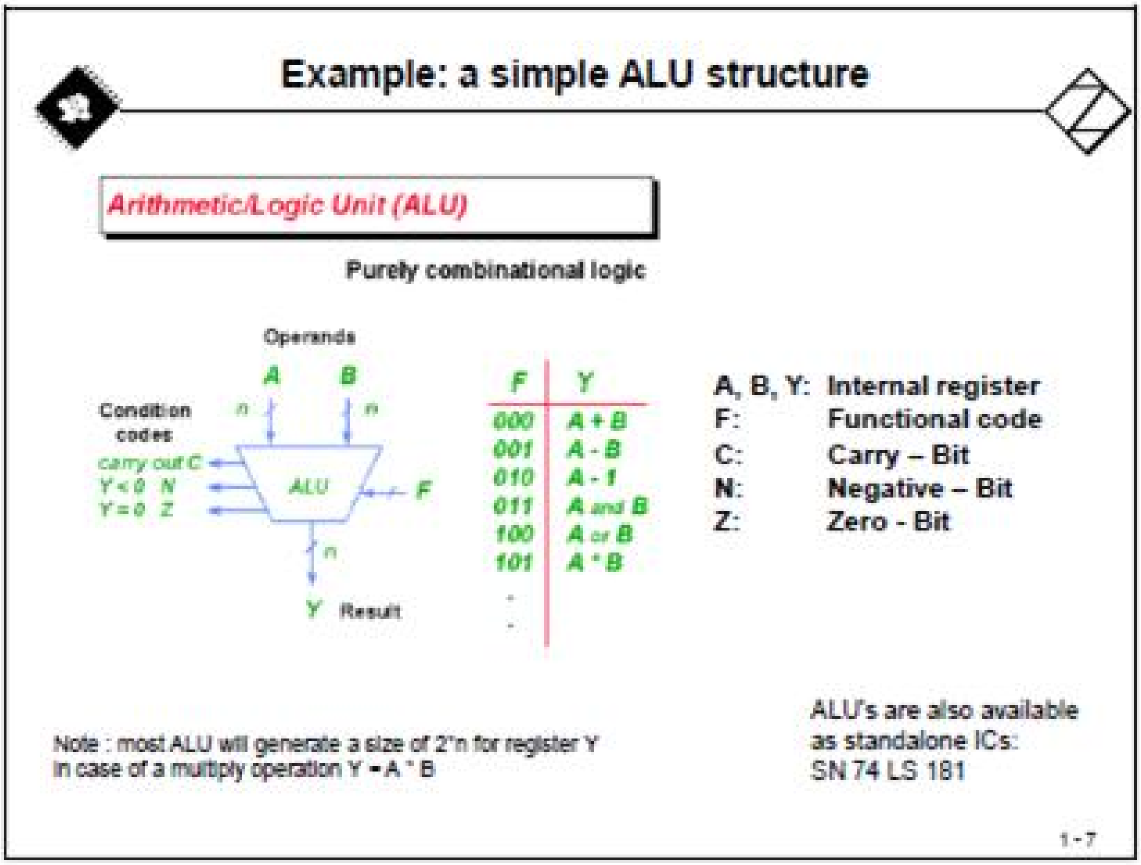
arithmetical:	Subtraction (SUB)
	Multiplication (MUL)
logical:	Comparison (CMP)
	Disjunction (OR)
	Antivalence (EXOR)
miscellaneous:	Right- and Left Shift (ASR,ASL)
	Rotation (ROL, ROR)
	Register-Bit-Manipulation (set, clear, toggle, test)

- a ALU is able to process two binary values with equal length (N) ➔ N-Bit ALU with N = 4,8,16,32 or 64
- most ALU's process Fixed Point Numbers
- A few ALU's, used especially in Digital Signal Processors and desktop processors, are capable to operate on Floating Point Numbers or on both formats.

1 - 6

Figure 7

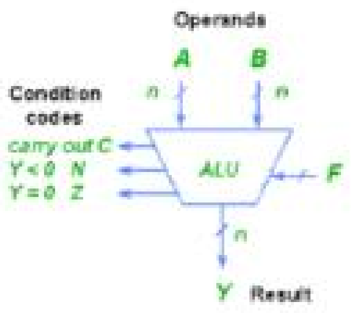
An ALU performs the arithmetic and logic operations of which the microprocessor is capable. A minimal requirement for an ALU is to perform ADD, NEG and AND. Other operations shown in the slide above, improve the performance of a specific microprocessor. A virtual ALU could look like this:



Example: a simple ALU structure

Arithmetic/Logic Unit (ALU)

Purely combinational logic



F	Y
000	A + B
001	A - B
010	A - 1
011	A and B
100	A or B
101	A * B
.	
.	

A, B, Y:	Internal register
F:	Functional code
C:	Carry – Bit
N:	Negative – Bit
Z:	Zero - Bit

Note : most ALU will generate a size of 2*n for register Y in case of a multiply operation Y = A * B

ALU's are also available as standalone ICs: SN 74 LS 181

1 - 7

Figure 8

The Intel 80x86: the legacy microprocessor

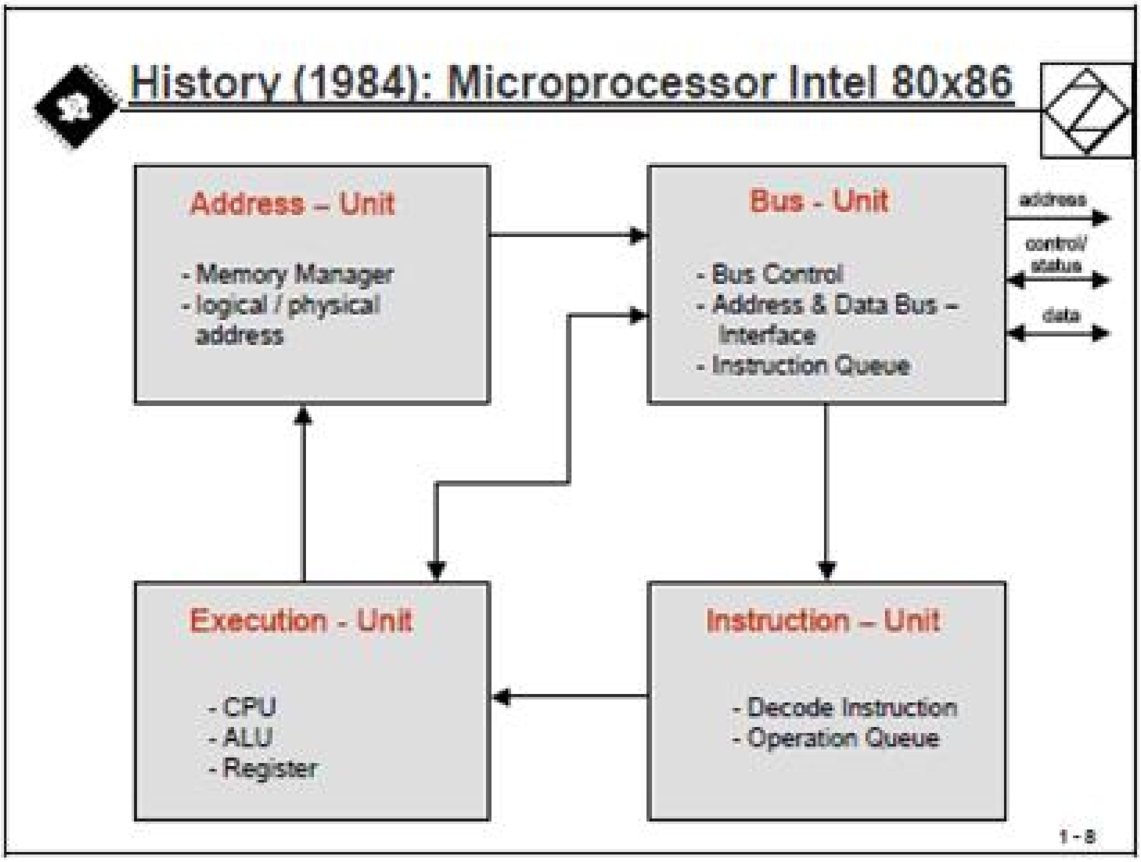


Figure 9

The Intel 8086 can be considered to be the veteran of all 16-bit microprocessors. Inside this processor four units manage the sequence of states. The bus-unit is responsible for addressing the external memory resources using a group of unidirectional digital address signals, bi-directional data lines, and control and status signals. Its purpose is to fill a first pipeline, called the “Instruction queue” with the next machine instructions to be processed. It is controlled by the Execution Unit and the Address-Unit.

The Instruction Unit reads the next instruction out of the Instruction queue, decodes it, and fills a second queue, the “Operation queue”, with the next internal operations that must be performed by the Execution Unit.

The Execution Unit does the ‘real’ work; it executes operations or calls the Bus Unit to read an optional operand from memory.

Once an instruction is completed, the Execution Unit forces the Address Unit to generate the address of the next instruction. If this instruction was already loaded into the Instruction queue, the operational speed is increased. This principle is called a “cache”.

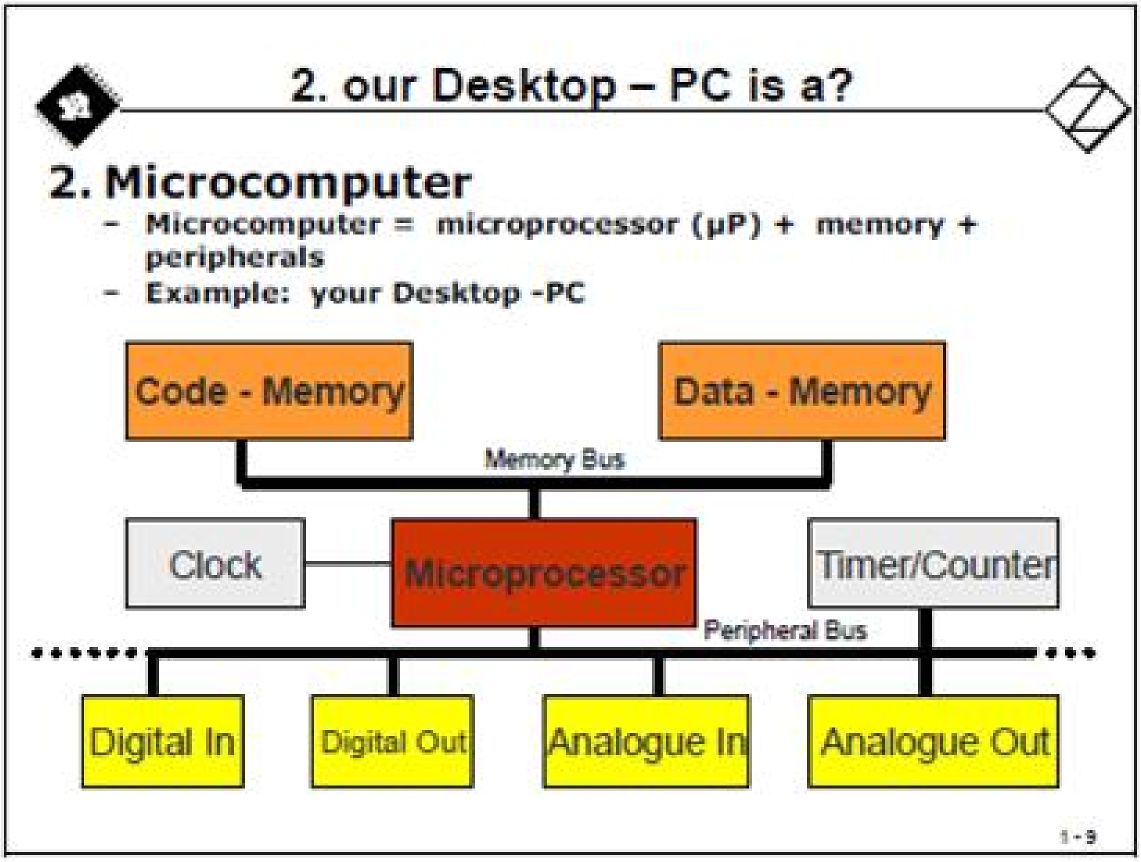
We could go much deeper into the workings of a Microprocessor; eventually you can book another class at your university that deals with this subject much more in detail, especially into the pros and cons of Harvard versus Von-Neumann Machines, into RISC versus CISC, versions of memory accesses etc.

For now, let us just keep in mind the basic operation of this type of device.

The Desktop - PC: a Microcomputer

When we add external devices to a microprocessor, we end up with the set-up for a computer system. We need to add external memory both for instructions ("code") and data to be computed. We also have to use some sort of connections to the outside world to our system. In general, they are grouped into digital input/outputs and analog input/outputs.

The following Slide (Figure 10) is a simplified block diagram of a typical microcomputer. As you can imagine, the latest designs of microcomputers are much more complex and are equipped with a lot more hierarchical levels. To keep it simple, let us focus on such a simplified architecture first.



2. our Desktop – PC is a?

2. Microcomputer
- Microcomputer = microprocessor (µP) + memory + peripherals
- Example: your Desktop -PC

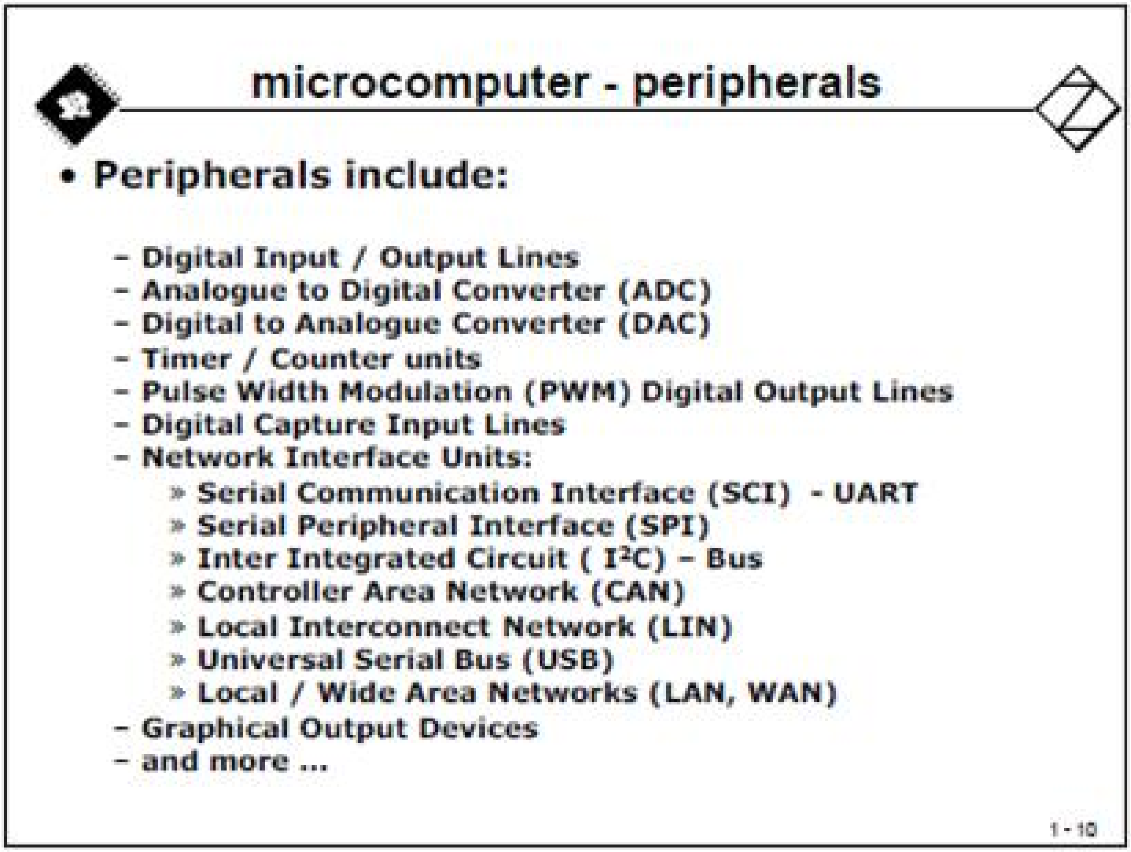
Figure 10

In general, the microprocessor in a microcomputer is connected to the memory system via a "memory" bus. In "von -Neumann" – microprocessor architectures this bus is shared between code and program memory, whereas in "Harvard" – microprocessor architectures, this bus system is separated into two independent, parallel bus systems.

The "Peripheral" bus in the slide above connects the microprocessor to units, which allow the processor to communicate with its environment (sensors, actuators, communication lines etc.). Some microcomputers use a dedicated "Peripheral" bus, as shown in Slide 1-9, whereas other devices integrate these peripheral functions into their data memory and call them "Data Memory Mapped Peripherals".

Microcomputer Peripherals

The following slide (Figure 11) is a non-exclusive list of some of the peripheral functions of a microcomputer. Peripherals are the interface of a microcomputer to the "real world". These units allow a microcomputer to communicate with sensors and actuators to exchange data and information with other nodes through network interface units.



microcomputer - peripherals

- Peripherals include:
  - Digital Input / Output Lines
  - Analogue to Digital Converter (ADC)
  - Digital to Analogue Converter (DAC)
  - Timer / Counter units
  - Pulse Width Modulation (PWM) Digital Output Lines
  - Digital Capture Input Lines
  - Network Interface Units:
    - » Serial Communication Interface (SCI) - UART
    - » Serial Peripheral Interface (SPI)
    - » Inter Integrated Circuit (I^2C) – Bus
    - » Controller Area Network (CAN)
    - » Local Interconnect Network (LIN)
    - » Universal Serial Bus (USB)
    - » Local / Wide Area Networks (LAN, WAN)
  - Graphical Output Devices
  - and more ...

1 - 10

Figure 11

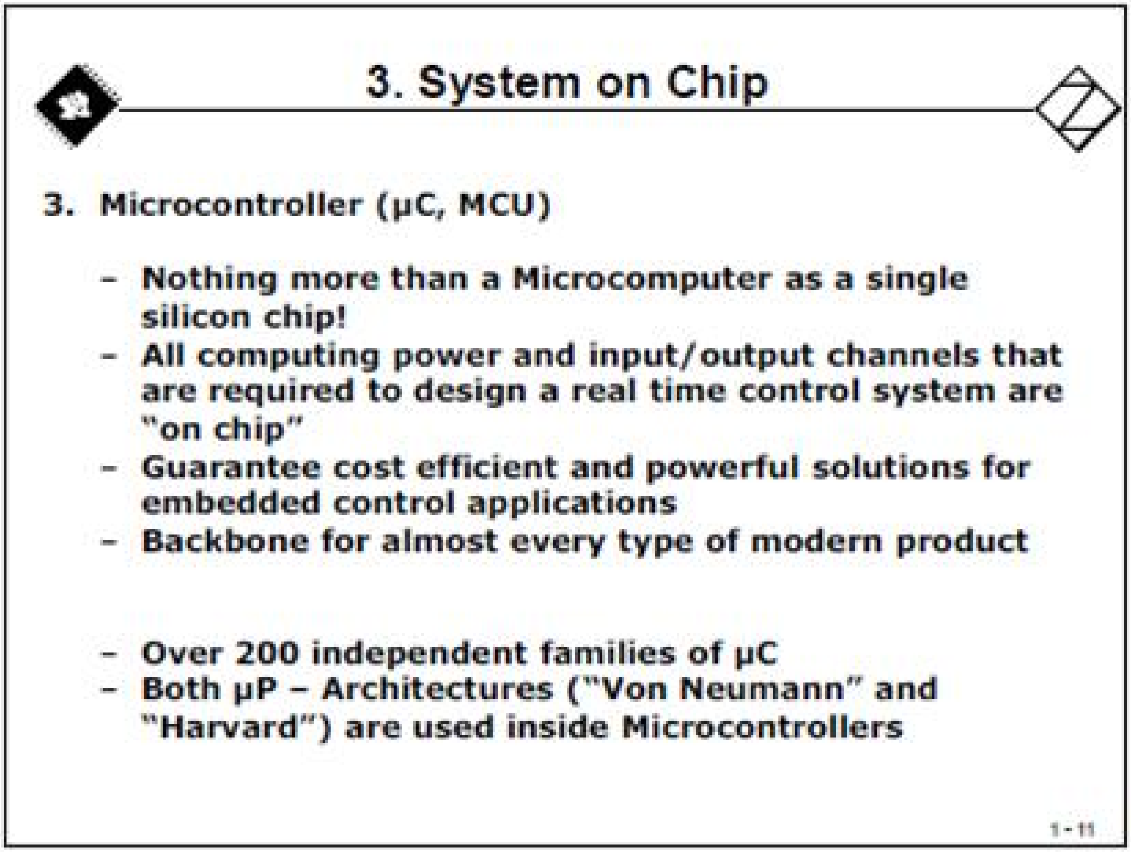
Modern microcomputers are equipped with a lot of enhanced peripheral units. To keep it simple, let us focus on basic peripheral units here. If you are familiar with microcomputers and you like to work with such hardware units, you can easily inspect all those fascinating peripheral units on a state of the art microcomputer.

The Microcontroller: a single chip computer

As technology advances, we want the silicon industry to build everything that is necessary for a microcomputer into a single piece of silicon, and we would end up with a microcontroller ("μC"). Of course nobody will try to include every single peripheral that is available or thinkable into a single chip – because nobody can afford to buy this "monster"-chip. On the contrary, engineers demand a microcontroller that suits their applications best but costs the least. This leads to a huge number of dedicated microcontroller families with totally different internal units, instruction sets, number of peripherals, and internal memory spaces. No customer will ask for a microcontroller with an internal code memory size of 16 megabytes, if the application fits easily into 64 kilobytes.

Today, microcontrollers are built into almost every industrial product that is available on the market. Try to guess how many microcontrollers you possess at home! The problem is you cannot see them from outside the product. That is the reason why they are also called "embedded" computers or "embedded" controllers. A sophisticated product such as the modern car is equipped with up to 80 microcontrollers to execute all the new electronic functions like antilock braking system (ABS), electronic stability program (ESP), adaptive cruise control (ACC), central locking, electrical mirror and seat adjustments, etc. On the other hand, a simple device such as a vacuum cleaner is equipped with a microcontroller to control the speed of the motor and the filling state of the cleaner. Not to speak of the latest developments in vacuum cleaner electronics: the cleaning robot with lots of control and sensor units to do the housework – with a much more powerful μC of course.

Microcontrollers are available as 4, 8, 16, 32 or even 64-bit devices, with each number showing the number of bits of an operand can process in parallel. If a microcontroller is a 32-bit type, the internal data memory is connected to the core unit with 32 internal signal lines.



3. System on Chip

3. Microcontroller (µC, MCU)

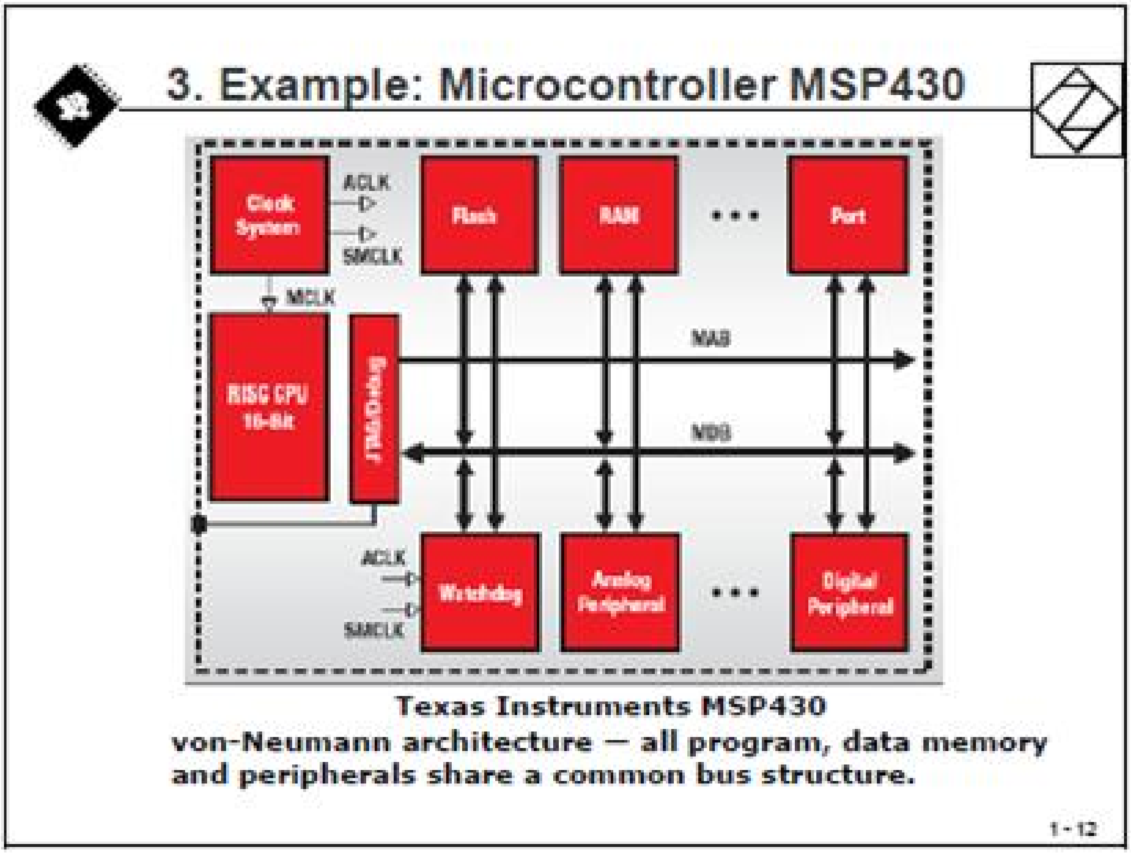
- Nothing more than a Microcomputer as a single silicon chip!
- All computing power and input/output channels that are required to design a real time control system are "on chip"
- Guarantee cost efficient and powerful solutions for embedded control applications
- Backbone for almost every type of modern product

- Over 200 independent families of µC
- Both µP – Architectures ("Von Neumann" and "Harvard") are used inside Microcontrollers

1 - 11

Figure 12

The MSP430 - a typical microcontroller



3. Example: Microcontroller MSP430

Texas Instruments MSP430
von-Neumann architecture — all program, data memory and peripherals share a common bus structure.

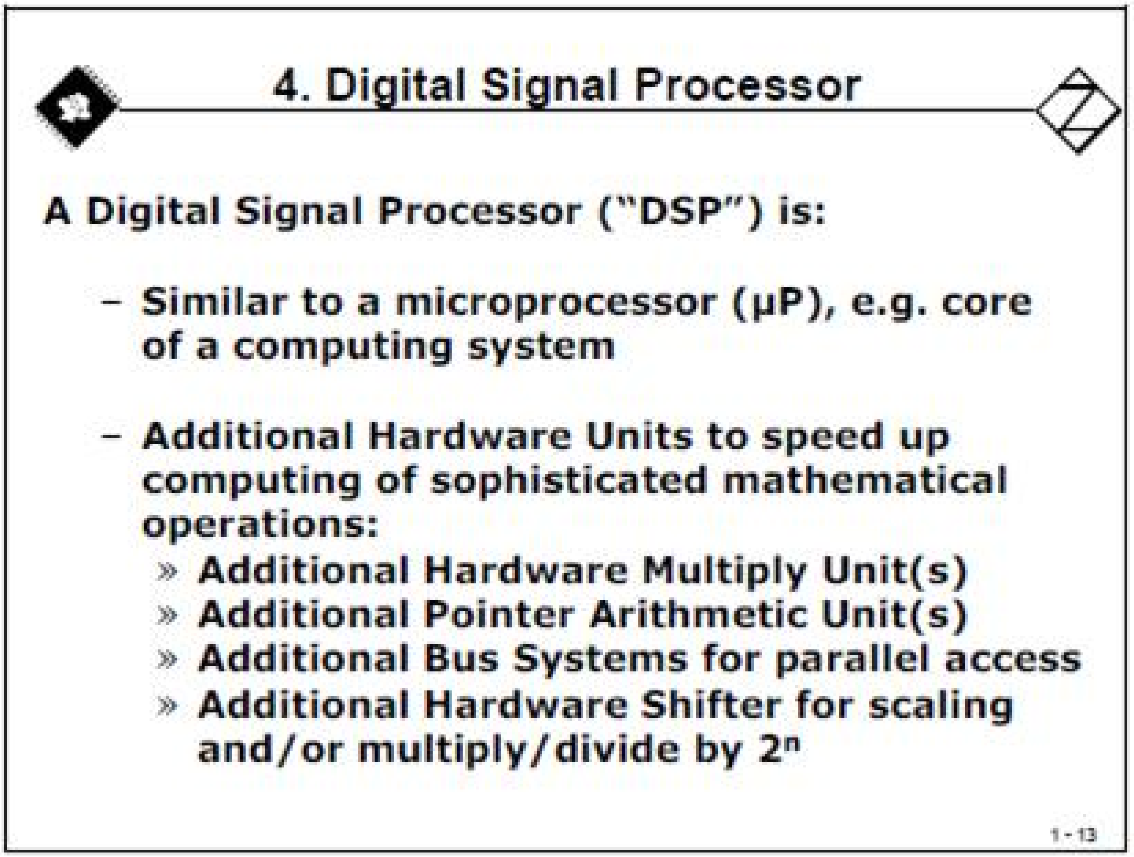
Figure 13

There are hundreds of types of micro controllers in the highly competitive market of embedded systems. They all have their pros and cons. Depending on the application area, budget limitations, and project requirements, one has to decide which one is best suited to fulfill the need. The slide above shows a block diagram of one of the most power effective micro controllers in the market – the MSP430. It comes with integrated memory blocks – FLASH for non - volatile storage of code sequences and RAM to store variables and results. It is equipped with internal analog and digital peripherals as well as communication channels.

The MSP430 family contains much more enhanced versions as shown in the block diagram in Figure 13. Some members of this family have integrated LCD display drivers, hardware multipliers or direct memory access (DMA) units, just to name a few. If you are more interested in that family, please use the corresponding Texas Instruments Teaching CD-ROM for that family.

A Digital Signal Processor

A Digital Signal Processor is a specific device that is designed around the typical mathematical operations to manipulate digital data that are measured by signal sensors. The objective is to process the data as quickly as possible to be able to generate an output stream of ‘new’ data in “real time”.



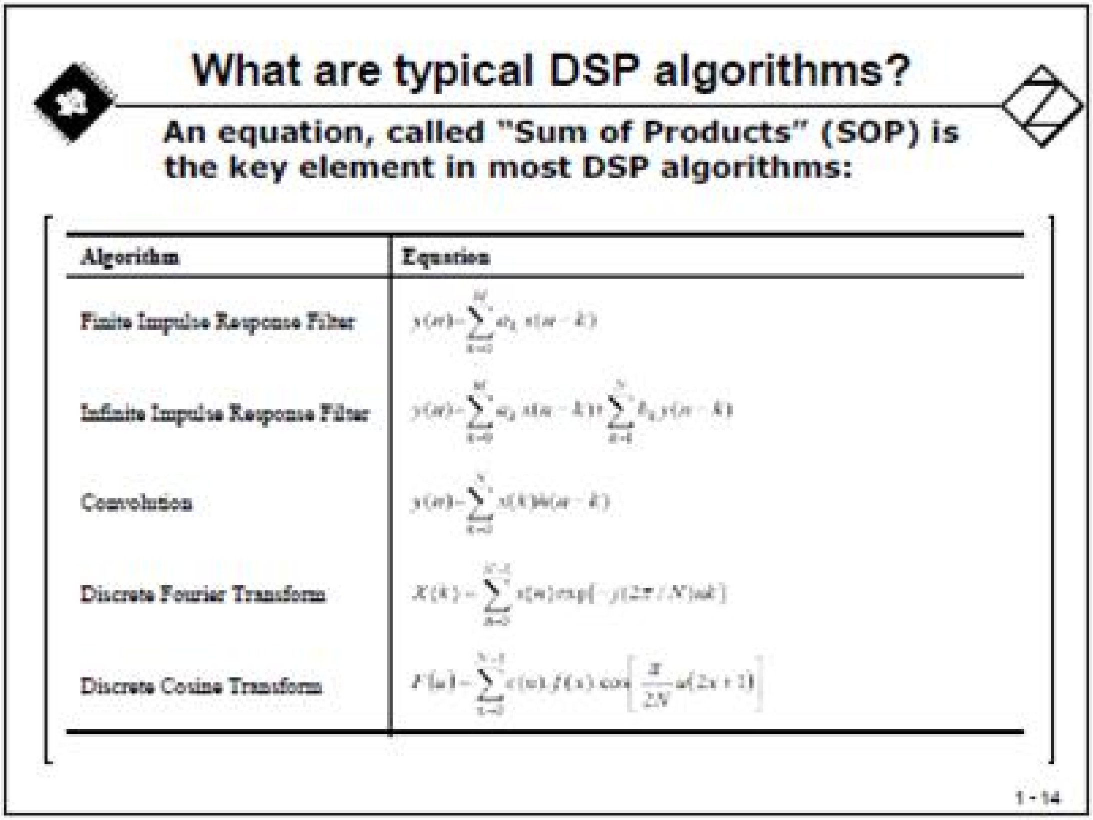
4. Digital Signal Processor

A Digital Signal Processor ("DSP") is:

- Similar to a microprocessor (µP), e.g. core of a computing system
- Additional Hardware Units to speed up computing of sophisticated mathematical operations:
  - » Additional Hardware Multiply Unit(s)
  - » Additional Pointer Arithmetic Unit(s)
  - » Additional Bus Systems for parallel access
  - » Additional Hardware Shifter for scaling and/or multiply/divide by 2^n

1 - 13

Figure 14



What are typical DSP algorithms?

An equation, called "Sum of Products" (SOP) is the key element in most DSP algorithms:

Algorithm	Equation
Finite Impulse Response Filter	$y(n)=\sum_{k=0}^{M} a_k\, x(n-k)$
Infinite Impulse Response Filter	$y(n)=\sum_{k=0}^{M} a_k\, x(n-k)+\sum_{k=1}^{N} b_k\, y(n-k)$
Convolution	$y(n)=\sum_{k=0}^{N} x(k)h(n-k)$
Discrete Fourier Transform	$X(k)=\sum_{n=0}^{N-1} x(n)\exp[-j(2\pi/N)nk]$
Discrete Cosine Transform	$F(u)=\sum_{x=0}^{N-1} c(u).f(x).\cos\left[\frac{\pi}{2N}u(2x+1)\right]$

1 - 14

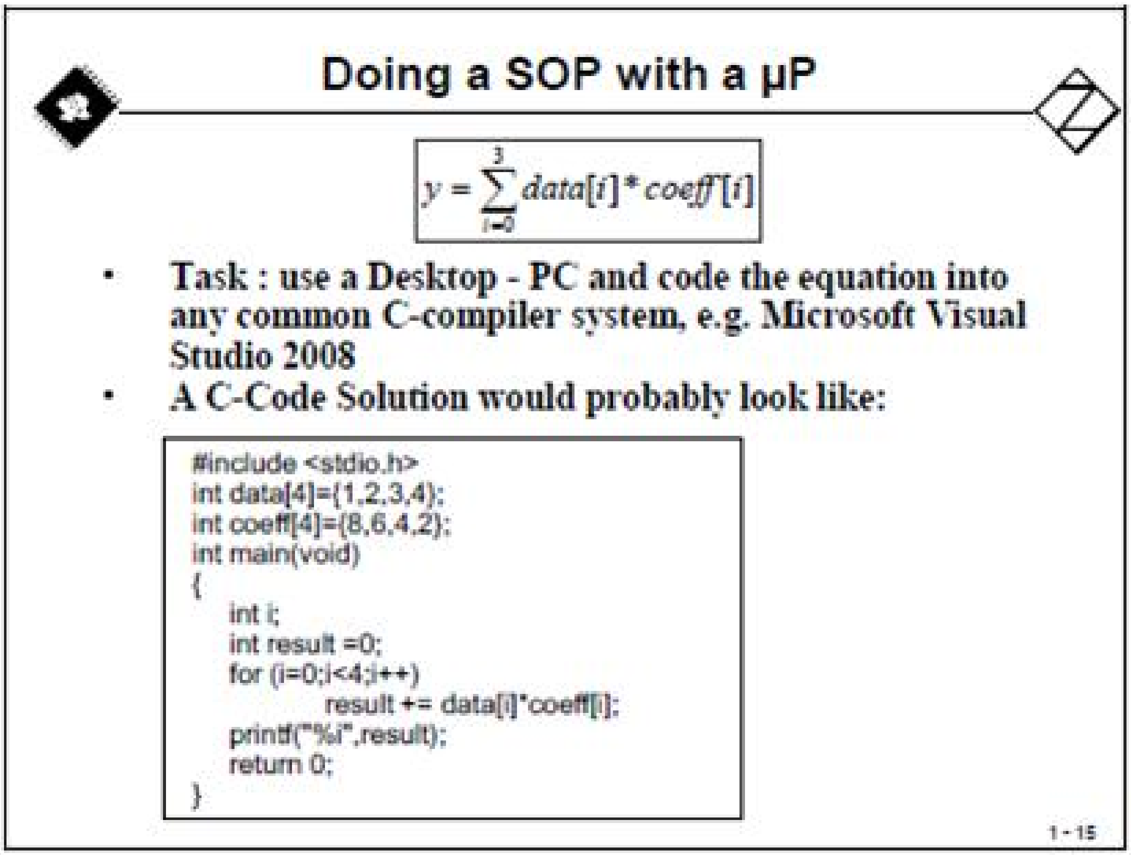
Figure 15

The "Sum of Product" - Equation
We won't go into the details of the theory of Digital Signal Processing now. Again, look out for additional classes at your university to learn more about the math behind this amazing part of modern technology. I highly recommend it. It is not the easiest topic, but it is worth it. Consider a future world without anybody who understands how a mobile phone or an autopilot of an airplane works internally –it's a terrible thought.

To begin with, let us scale down all the math into one basic equation that is behind almost all approaches of Digital Signal Processing. It is the "Sum of Products" formula. A new value "y" is calculated as a sum of partial products. Two arrays "data" and "coeff" are multiplied as pairs and the products are added together. Depending on the data type of the input arrays we could solve this equation in floating point or integer mathematics. Integer is most often also called "fixed - point" math (see Chapter 2).

In contrast to its predecessor, the TMS320F28335 is both a floating-point and also fixed-point device, so we can use the best of both worlds. To keep it simple for now, let's stay with fixed - point mathematics first. In chapter 2 we will discuss the pros and cons of fixed point versus floating point DSPs in a little bit more depth.

In a standard ANSI-C we can easily define two arrays of integer input data and the code lines that are needed to calculate the output value "y":



Doing a SOP with a µP

$$y = \sum_{i=0}^{3} data[i] * coeff[i]$$

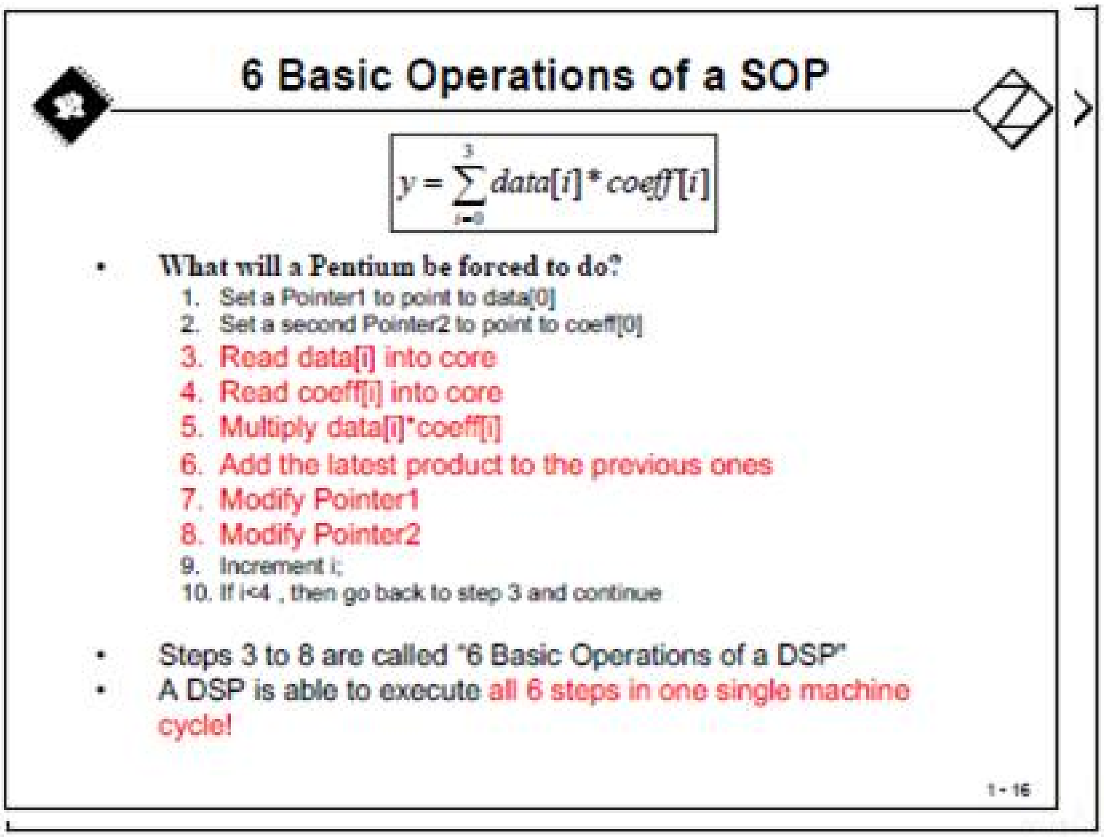
- Task : use a Desktop - PC and code the equation into any common C-compiler system, e.g. Microsoft Visual Studio 2008
- A C-Code Solution would probably look like:

```
#include <stdio.h>
int data[4]={1,2,3,4};
int coeff[4]={8,6,4,2};
int main(void)
{
    int i;
    int result =0;
    for (i=0;i<4;i++)
            result += data[i]*coeff[i];
    printf("%i",result);
    return 0;
}
```

1 - 15

Figure 16

If we look a little bit more in detail into the tasks that needs to be solved by a standard processor we can distinguish 10 steps. Due to the sequential nature of this type of processor, it can do only one of these 10 steps at a time. This will consume a considerable amount of the computing power of this processor. For our tiny example, the processor must loop between step 3 and step 10 a total of four times. For real Digital Signal Processing the SOP – procedure is going to much higher loop repetitions – forcing the standard processor to use even more computing power.



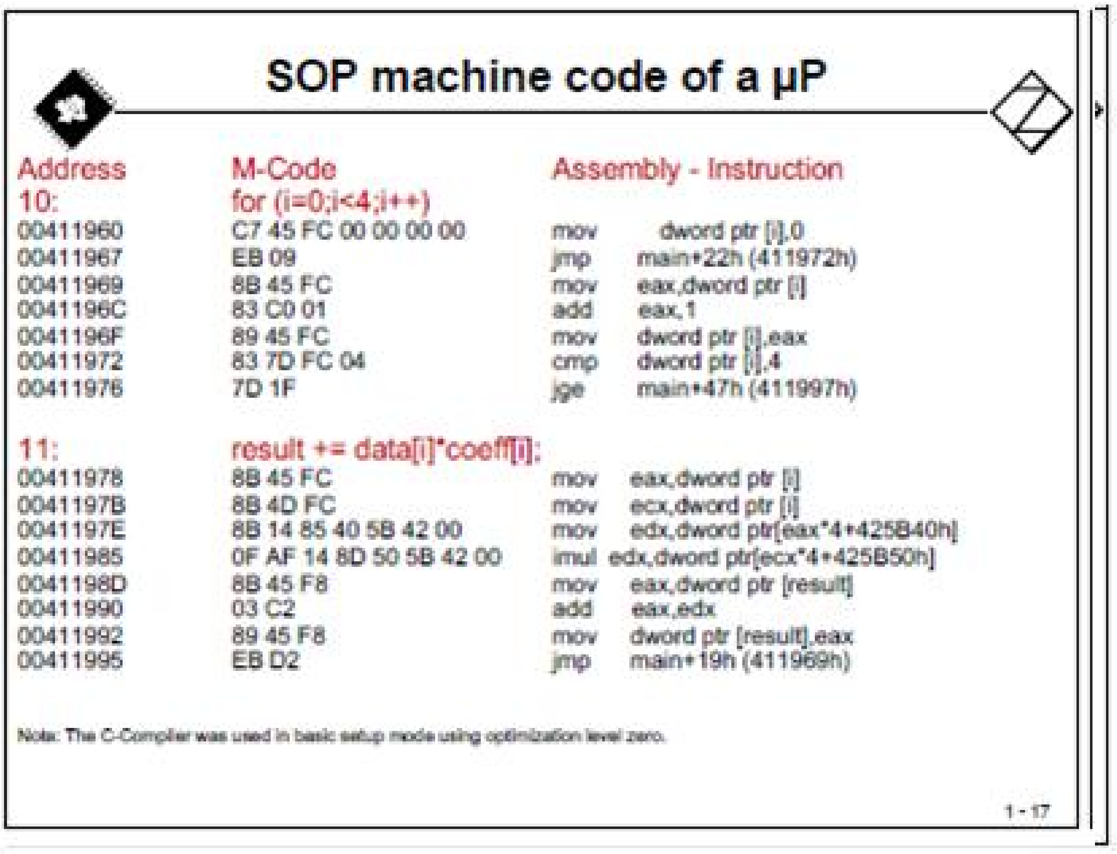
6 Basic Operations of a SOP

$$\boxed{y = \sum_{i=0}^{3} data[i] * coeff[i]}$$

- What will a Pentium be forced to do?
  1. Set a Pointer1 to point to data[0]
  2. Set a second Pointer2 to point to coeff[0]
  3. Read data[i] into core
  4. Read coeff[i] into core
  5. Multiply data[i]*coeff[i]
  6. Add the latest product to the previous ones
  7. Modify Pointer1
  8. Modify Pointer2
  9. Increment i;
  10. If i<4 , then go back to step 3 and continue
- Steps 3 to 8 are called "6 Basic Operations of a DSP"
- A DSP is able to execute all 6 steps in one single machine cycle!

1 - 16

Figure 17



SOP machine code of a µP

```
Address     M-Code                        Assembly - Instruction
10:         for (i=0;i<4;i++)
00411960    C7 45 FC 00 00 00 00          mov     dword ptr [i],0
00411967    EB 09                         jmp     main+22h (411972h)
00411969    8B 45 FC                      mov     eax,dword ptr [i]
0041196C    83 C0 01                      add     eax,1
0041196F    89 45 FC                      mov     dword ptr [i],eax
00411972    83 7D FC 04                   cmp     dword ptr [i],4
00411976    7D 1F                         jge     main+47h (411997h)

11:         result += data[i]*coeff[i];
00411978    8B 45 FC                      mov     eax,dword ptr [i]
0041197B    8B 4D FC                      mov     ecx,dword ptr [i]
0041197E    8B 14 85 40 5B 42 00          mov     edx,dword ptr[eax*4+425B40h]
00411985    0F AF 14 8D 50 5B 42 00       imul    edx,dword ptr[ecx*4+425B50h]
0041198D    8B 45 F8                      mov     eax,dword ptr [result]
00411990    03 C2                         add     eax,edx
00411992    89 45 F8                      mov     dword ptr [result],eax
00411995    EB D2                         jmp     main+19h (411969h)
```

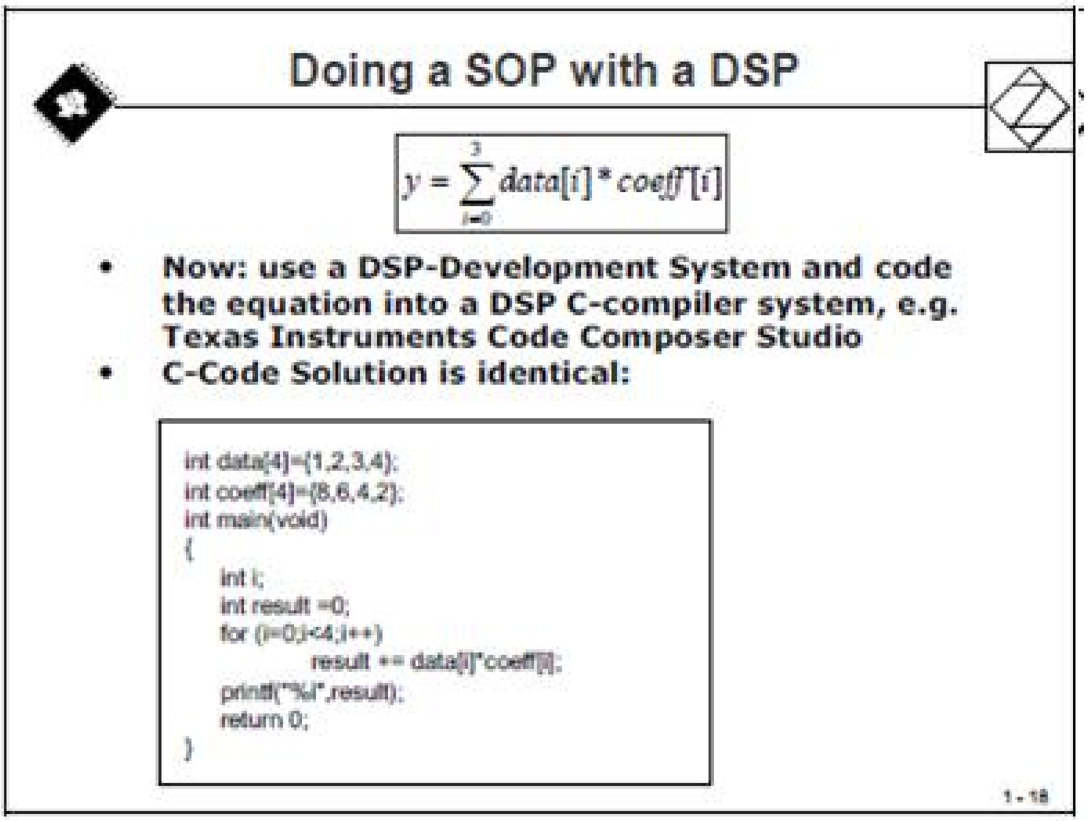
Note: The C-Compiler was used in basic setup mode using optimization level zero.

1 - 17

Figure 18

A SOP executed by a DSP

If we apply the SOP-task to a Digital Signal Processor of fixed-point type the ANSI-C code looks identical to the standard processor one. The difference is the output of the compilation! When you compare Figure 20 with Figure 18 you will notice the dramatic reduction in the consumption of the memory space and number of execution cycles. A DSP is much more appropriate to calculate a SOP in real time! Ask your professor about the details of the two slides.



Doing a SOP with a DSP

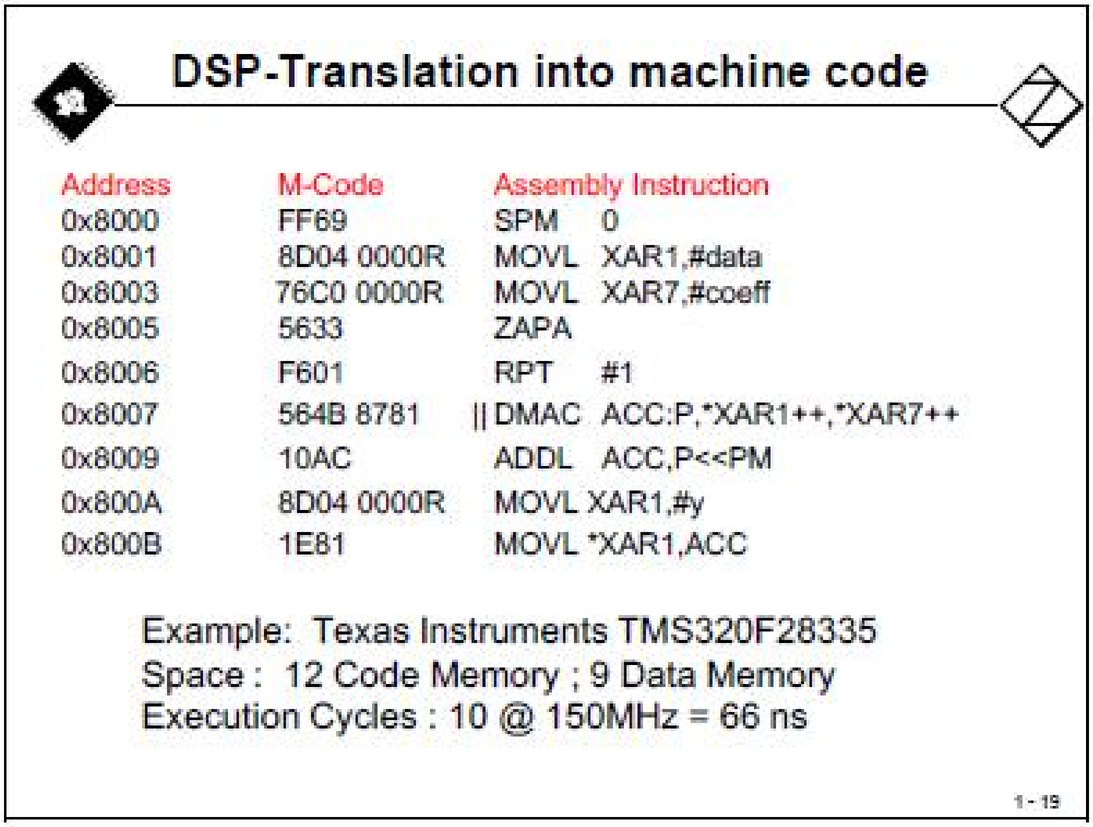
$$y = \sum_{i=0}^{3} data[i] * coeff[i]$$

- Now: use a DSP-Development System and code the equation into a DSP C-compiler system, e.g. Texas Instruments Code Composer Studio
- C-Code Solution is identical:

```
int data[4]={1,2,3,4};
int coeff[4]={8,6,4,2};
int main(void)
{
    int i;
    int result =0;
    for (i=0;i<4;i++)
              result += data[i]*coeff[i];
    printf("%i",result);
    return 0;
}
```

1 - 18

Figure 19



DSP-Translation into machine code

Address	M-Code	Assembly Instruction
0x8000	FF69	SPM 0
0x8001	8D04 0000R	MOVL XAR1,#data
0x8003	76C0 0000R	MOVL XAR7,#coeff
0x8005	5633	ZAPA
0x8006	F601	RPT #1
0x8007	564B 8781	|| DMAC ACC:P,*XAR1++,*XAR7++
0x8009	10AC	ADDL ACC,P<<PM
0x800A	8D04 0000R	MOVL XAR1,#y
0x800B	1E81	MOVL *XAR1,ACC

Example: Texas Instruments TMS320F28335
Space : 12 Code Memory ; 9 Data Memory
Execution Cycles : 10 @ 150MHz = 66 ns

1 - 19

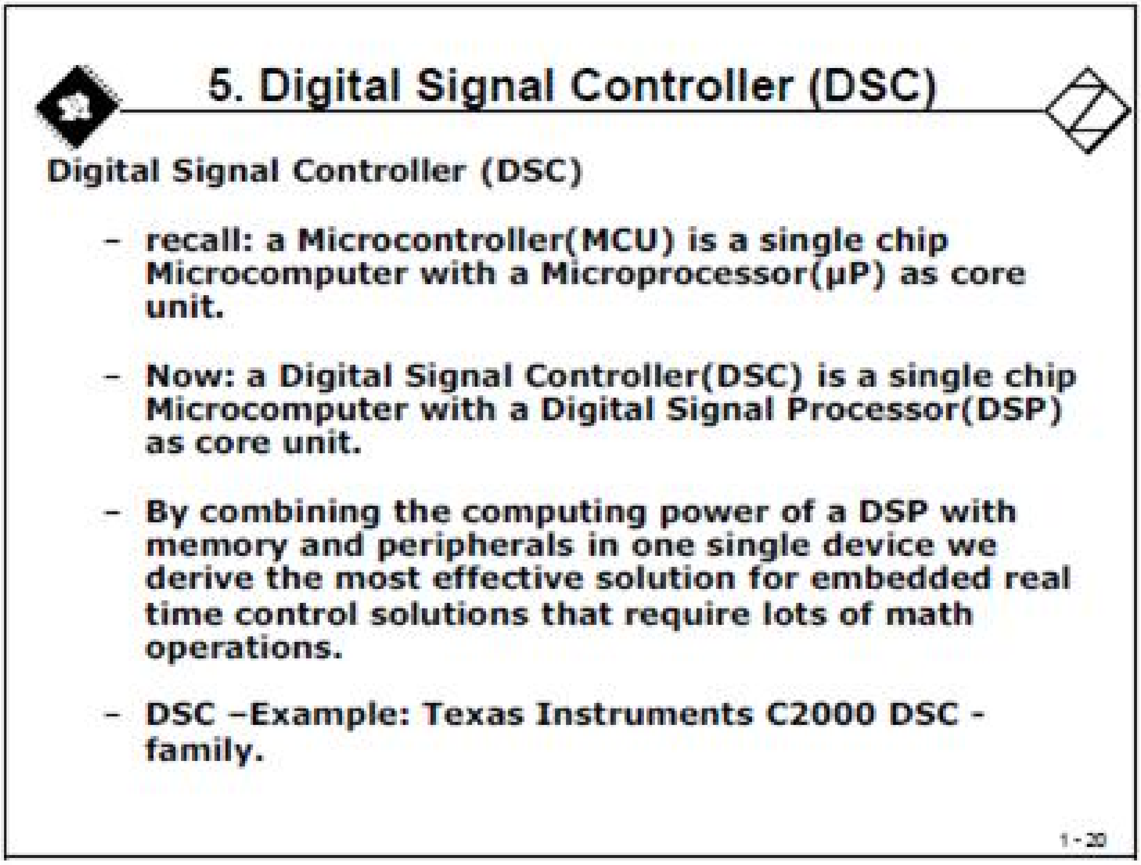
Figure 20

A Digital Signal Controller
Finally, a Digital Signal Controller (DSC) is a new type of microcontroller, where the processing power is delivered by a DSP – a single chip device combining both the computing power of a Digital Signal Processor and the embedded peripherals of a single chip computing system.

For advanced real time control systems with a high amount of mathematical calculations, a DSC is the best choice.

Today there are only a few manufacturers offering DSC's. Due to the advantages of DSC's for many projects, a number of silicon manufacturers are developing this type of controller.

This tutorial is based on the Texas Instruments TMS320F28335, a 32-bit floating point Digital Signal Controller (DSC).



5. Digital Signal Controller (DSC)

Digital Signal Controller (DSC)

- recall: a Microcontroller(MCU) is a single chip Microcomputer with a Microprocessor(µP) as core unit.
- Now: a Digital Signal Controller(DSC) is a single chip Microcomputer with a Digital Signal Processor(DSP) as core unit.
- By combining the computing power of a DSP with memory and peripherals in one single device we derive the most effective solution for embedded real time control solutions that require lots of math operations.
- DSC –Example: Texas Instruments C2000 DSC - family.

1 - 20

Figure 21

Note: Some manufacturers, like Infineon and Renesas, still call their DSCs microcontrollers. This is because most target applications are typically regarded as "microcontroller sockets" and many engineers are unfamiliar with the term DSC.

TI also recently changed the naming of the C2000 line from DSC to microcontroller.

DSP Competition

There are only a few global players in the area of DSP and DSC. As you can see from Figure 22 (for more details, go to: www.fwdconcepts.com), Texas Instruments is the absolute leader in this area. A working knowledge of TI-DSP will help you to master your professional career.

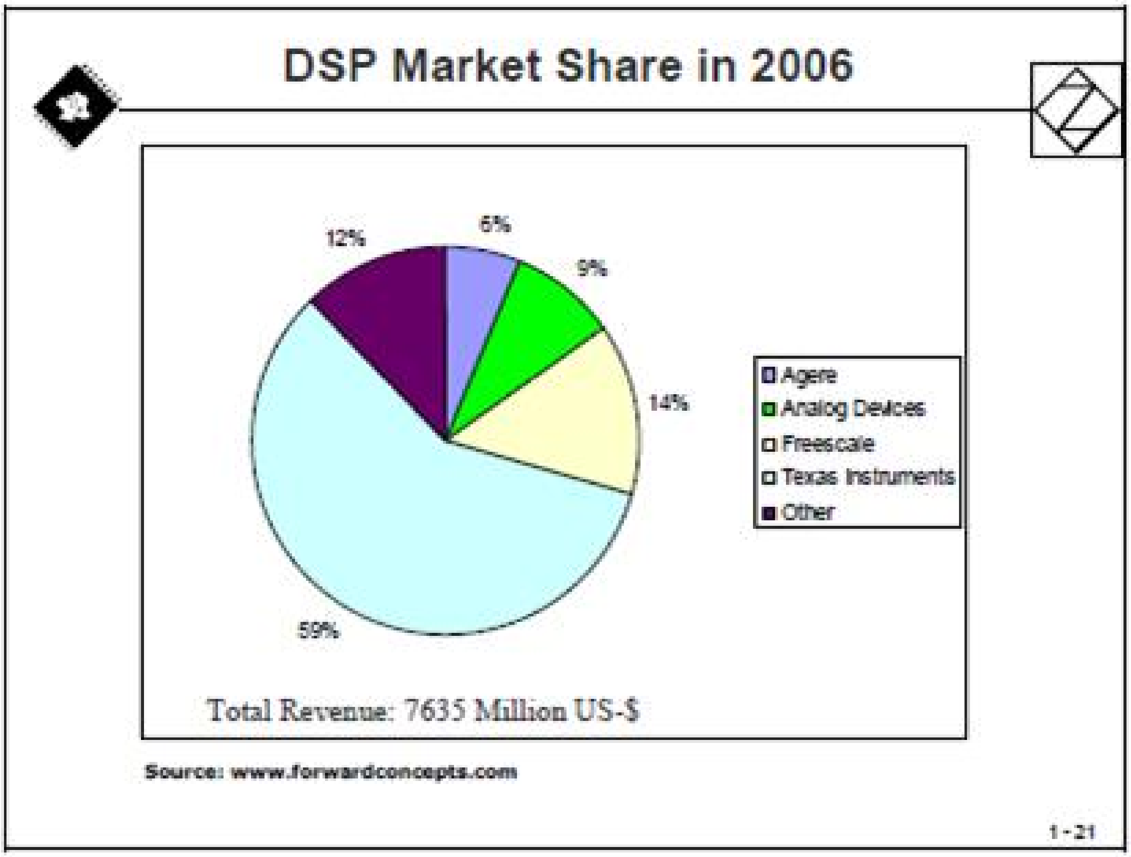


Figure 22

With such expertise in DSPs, it is only natural that the lessons TI has learned and technologies developed for DSPs trickle down also to TI's microcontrollers. As the leader in DSP, Texas Instruments microcontrollers will also challenge the market!

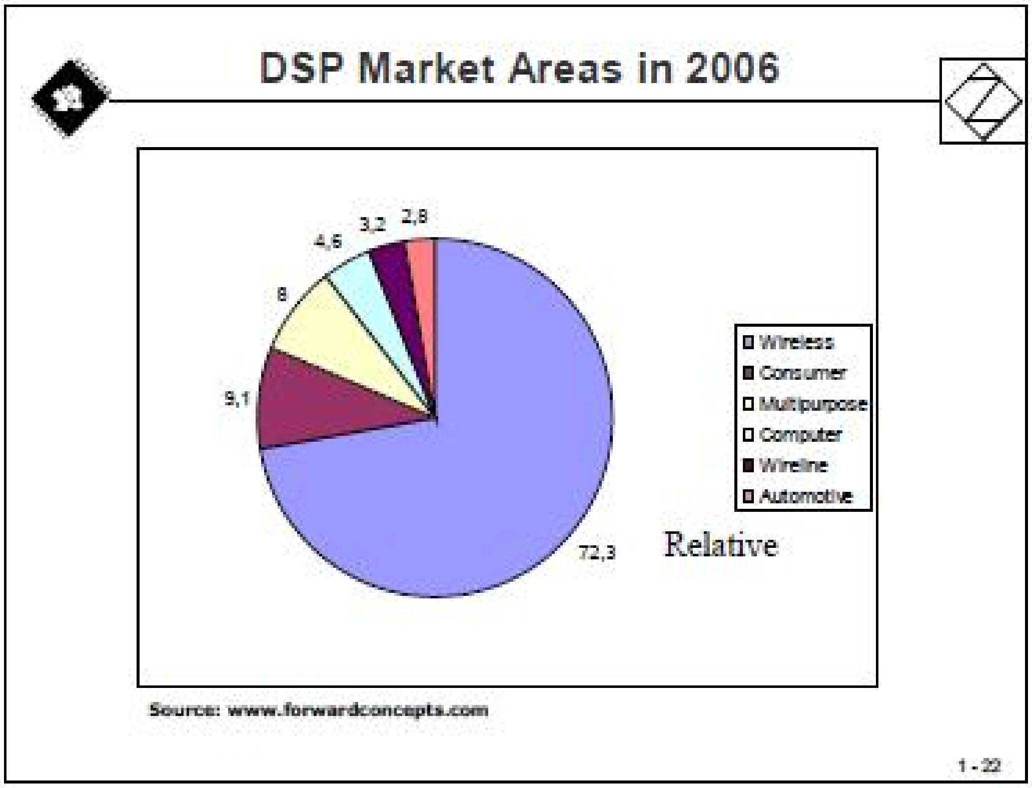
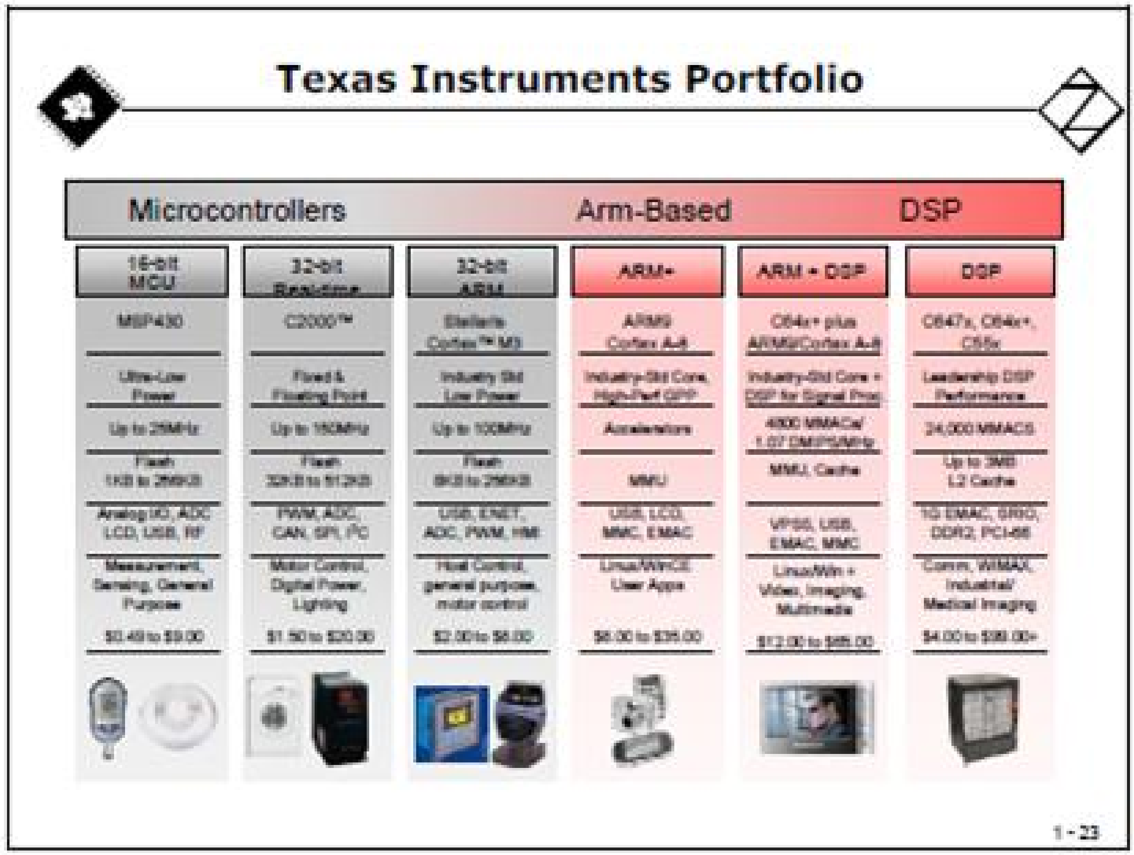


Figure 23

Texas Intruments DSP/DSC - Portfolio



Texas Instruments Portfolio

Microcontrollers			Arm-Based		DSP
16-bit MCU	32-bit Real-time	32-bit ARM	ARM+	ARM + DSP	DSP
MSP430	C2000™	Stellaris Cortex™ M3	ARM9 Cortex A-8	C64x+ plus ARM9/Cortex A-8	C647x, C64x+, C55x
Ultra-Low Power	Fixed & Floating Point	Industry Std Low Power	Industry-Std Core, High-Perf GPP	Industry-Std Core + DSP for Signal Proc.	Leadership DSP Performance
Up to 25MHz	Up to 150MHz	Up to 100MHz	Accelerators	4800 MMACs/ 1.07 DMIPS/MHz	24,000 MMACS
Flash 1KB to 256KB	Flash 32KB to 512KB	Flash 8KB to 256KB	MMU	MMU, Cache	Up to 3MB L2 Cache
Analog I/O, ADC LCD, USB, RF	PWM, ADC, CAN, SPI, I²C	USB, ENET, ADC, PWM, HMI	USB, LCD, MMC, EMAC	VPSS, USB, EMAC, MMC	1G EMAC, SRIO, DDR2, PCI-66
Measurement, Sensing, General Purpose	Motor Control, Digital Power, Lighting	Host Control, general purpose, motor control	Linux/WinCE User Apps	Linux/Win + Video, Imaging, Multimedia	Comm, WiMAX, Industrial/ Medical Imaging
$0.49 to $9.00	$1.50 to $20.00	$2.00 to $8.00	$8.00 to $35.00	$12.00 to $65.00	$4.00 to $99.00+

1 - 23

Figure 24

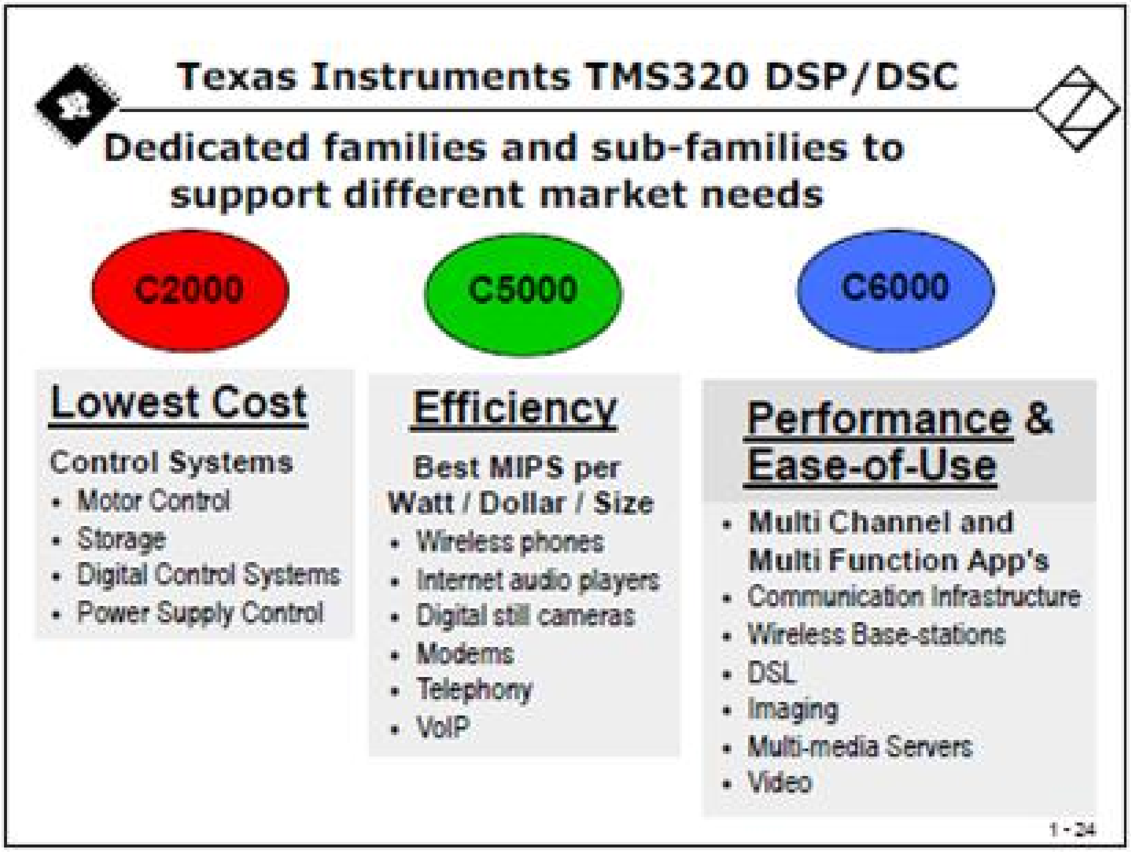
The DSP / DSC – portfolio of Texas instruments is split into three major device families, called Microcontrollers, ARM-based, and DSP.

The C64x branch is the most powerful series of DSP in computing power. There are floating – point as well as fixed – point devices in this family. The application fields are image processing, audio, multimedia server, base stations for wireless communication etc.

The C55x family is focused on mobile systems with very efficient power consumption per MIPS. Its main application area is in cell phone technology.

The C2000 – group is dedicated to Digital Signal Control (DSC), as you have learned from the first slides, and is a very powerful solution for real time control applications. This group is accompanied at the two ends by a 16-bit Microcontroller group (MSP430), and a 32-bit series of ARM-core based microcontrollers (Cortex M3, Cortex A-8 or ARM9).

Figure 25 summarizes the main application areas for the 3 Texas Instruments families of DSP.



Texas Instruments TMS320 DSP/DSC

Dedicated families and sub-families to support different market needs

C2000
C5000
C6000

Lowest Cost

Control Systems

- Motor Control
- Storage
- Digital Control Systems
- Power Supply Control

Efficiency

Best MIPS per Watt / Dollar / Size

- Wireless phones
- Internet audio players
- Digital still cameras
- Modems
- Telephony
- VoIP

Performance & Ease-of-Use

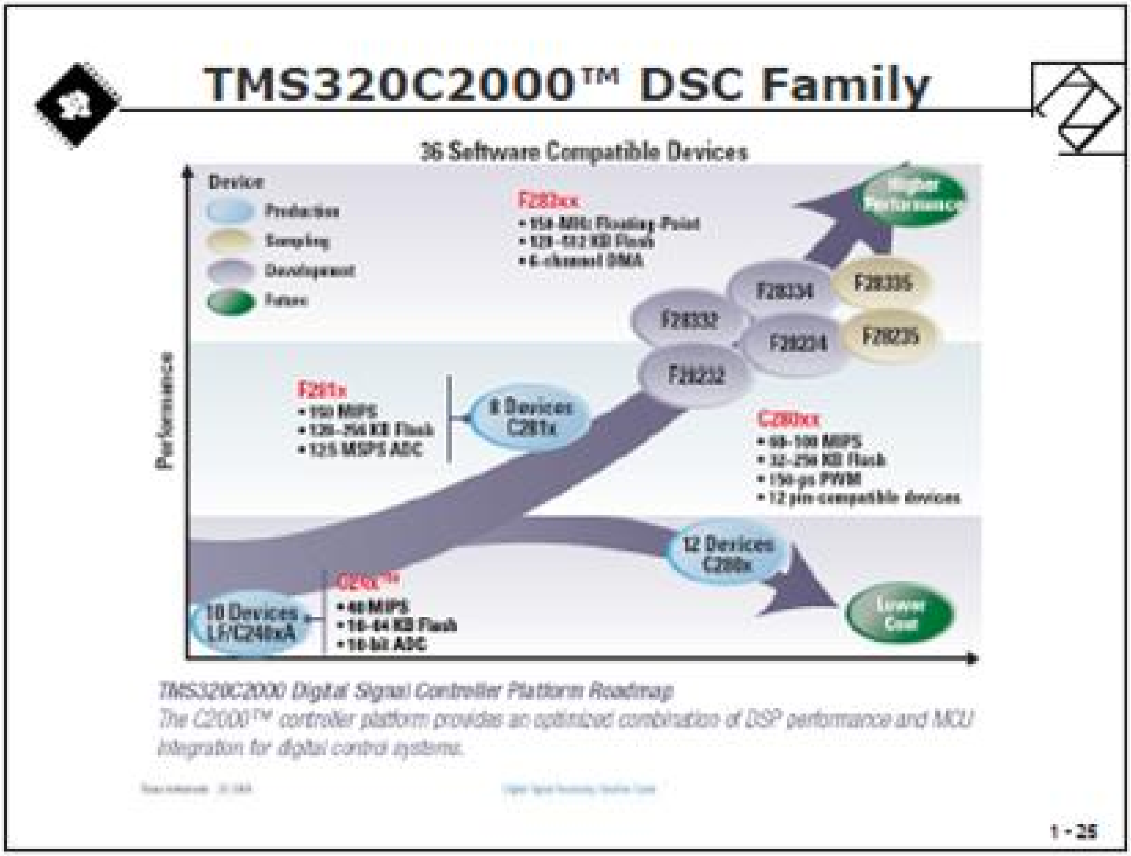
- Multi Channel and Multi Function App's
- Communication Infrastructure
- Wireless Base-stations
- DSL
- Imaging
- Multi-media Servers
- Video

1 - 24

Figure 25

TMS320F28x Roadmap

For the C2000 – family, we can distinguish between two groups of devices: a 16-bit group called TMS320C24x, and a 32-bit group called TMS320C28x.



TMS320C2000™ DSC Family

TMS320C2000 Digital Signal Controller Platform Roadmap
The C2000™ controller platform provides an optimized combination of DSP performance and MCU integration for digital control systems.

Figure 26

Figure 27 illustrates the latest developments in the 32-bit real-time controller family C28x:

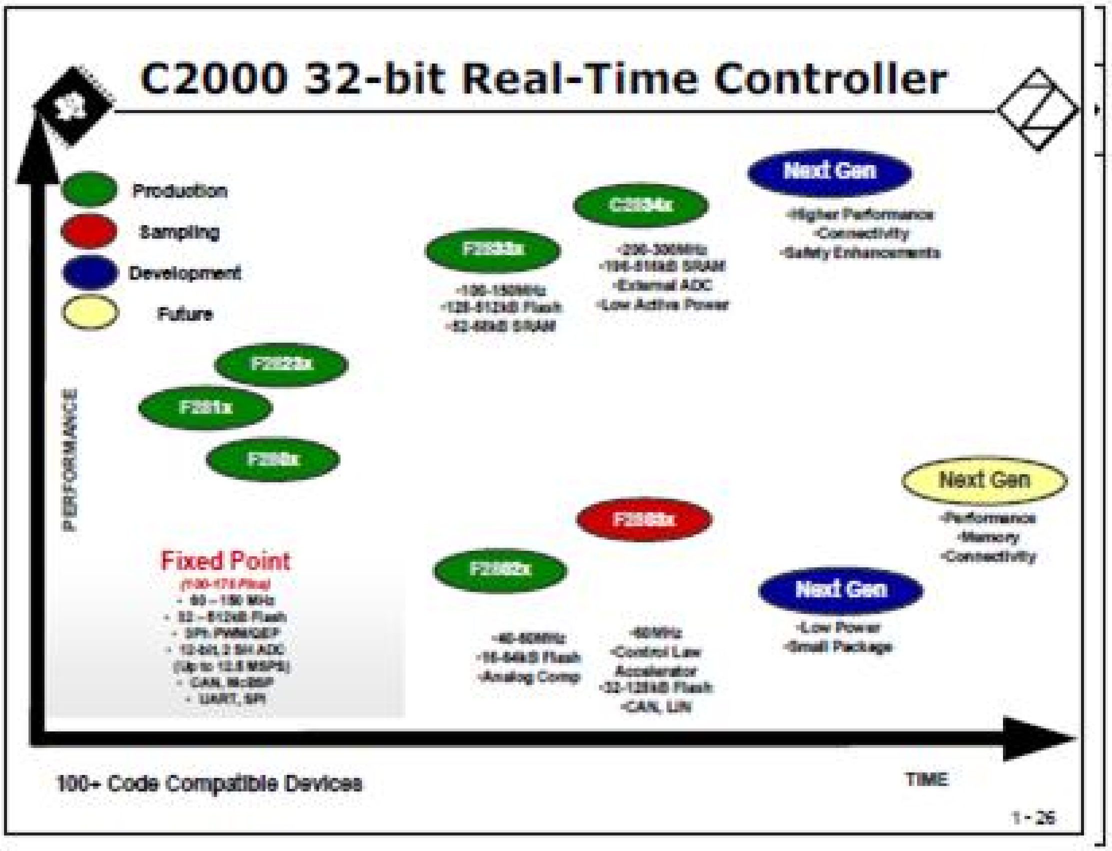

C2000 32-bit Real-Time Controller
Production
Sampling
Development
Future
PERFORMANCE
F2823x
F281x
F280x
F2833x
-100-150MHz
-128-512kB Flash
-52-68kB SRAM
C2834x
-200-300MHz
-196-516kB SRAM
-External ADC
-Low Active Power
Next Gen
-Higher Performance
-Connectivity
-Safety Enhancements
Fixed Point
(100-175 Pins)
- 60 – 150 MHz
- 32 – 512kB Flash
- 5Ph PWM/QEP
- 12-bit, 2 SH ADC
(Up to 12.5 MSPS)
- CAN, McBSP
- UART, SPI
F2802x
-40-60MHz
-16-64kB Flash
-Analog Comp
F2803x
-60MHz
-Control Law
Accelerator
-32-128kB Flash
-CAN, LIN
Next Gen
-Low Power
-Small Package
Next Gen
-Performance
-Memory
-Connectivity
100+ Code Compatible Devices
TIME
1 - 26

Figure 27

TMS320F28x Application Areas

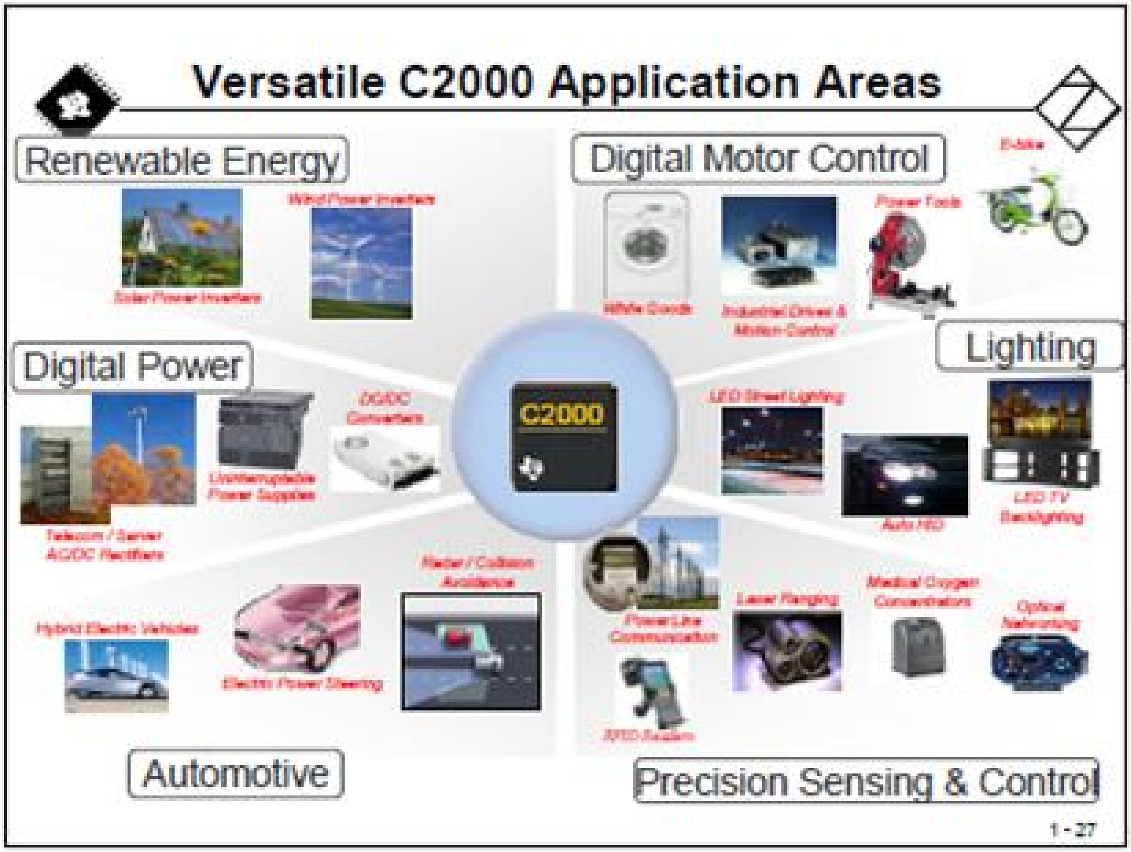


Figure 28

TMS320F28x Block Diagram

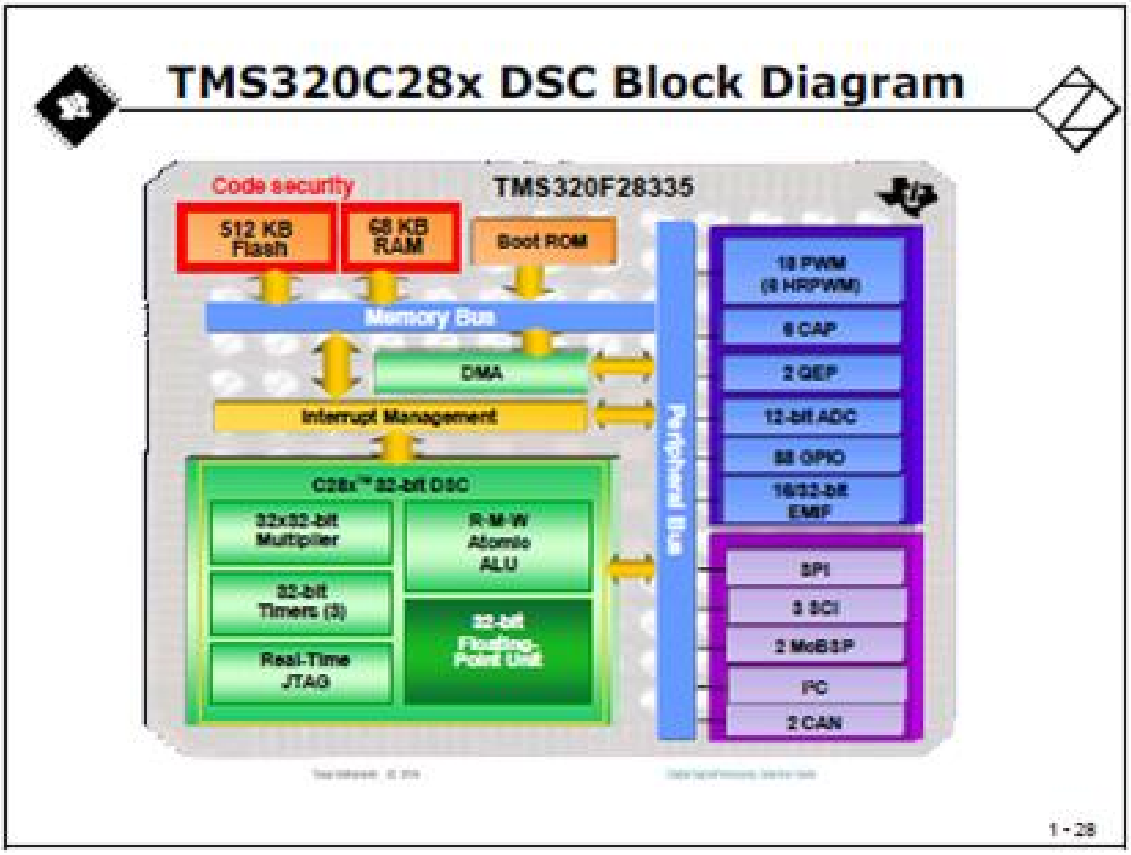


Figure 29