Introduction

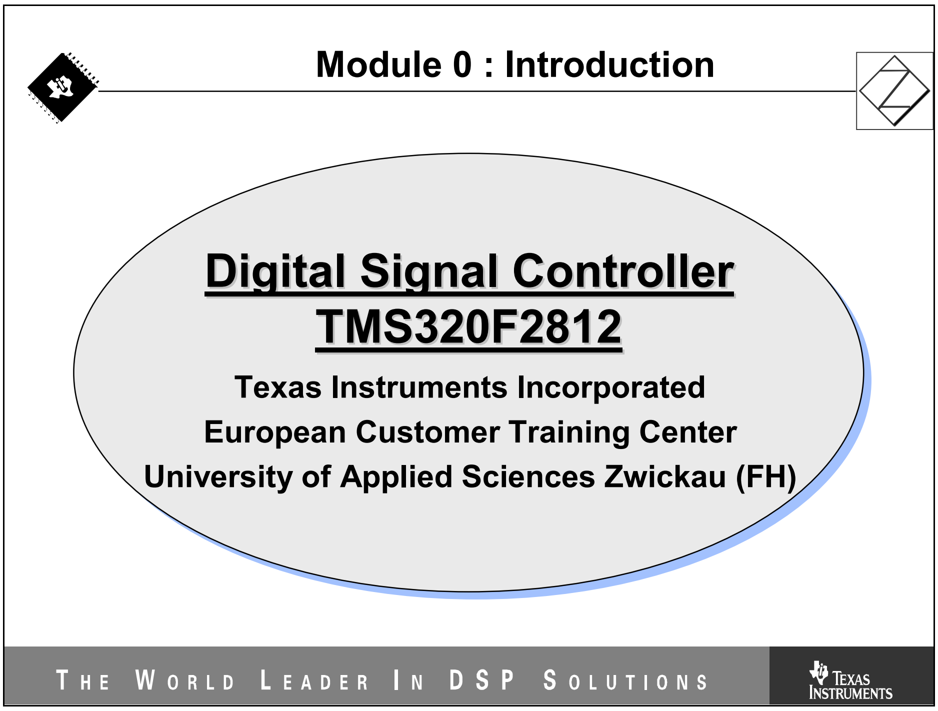
Welcome to the C28x - Tutorial

Welcome to the Texas Instruments TMS320F2812 Tutorial. This material is intended to be used as a student guide for a series of lessons and lab exercises dedicated to the TMS320F2812 Digital Signal Controller. The series of modules will guide you through the various elements of this device, as well as train you in using Texas Instruments development tools and additional resources from the Internet.

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 Part 1 (Modules 0 to 9) is to be used in one semester, accompanied by lab exercises in parallel. Each module includes a detailed lab procedure to be used by students during their lab sessions.

The experimental lab sessions are based on the eZdsp TMS320F2812, the Code Composer Studio IDE that is supplied with the eZdsp and some additional hardware (The "Zwickau Adapter Board"). Copies of this add-on board are available from the author. The schematic of the board is also part of this CD-ROM, so that you can build one yourself as well.

Part 2 (Modules 10 to 15) of the series goes deeper into details of the TMS320F2812. It covers more advanced subjects and can be seen as an optional series of lessons.



Module Topics

Modules Part I

Chapter 0: Introduction to DSP

Chapter 1: TMS320F2812 Architecture

Chapter 2: Software Development Tools

Chapter 3: Digital Input/Output

Chapter 4: Understanding the F2812 Interrupt System

Chapter 5: Event Manager

Chapter 6: Analogue to Digital Converter

Chapter 7: Communication I: Serial Peripheral Interface

Chapter 8: Communication II: Serial Communication Interface

Chapter 9: Communication III: Controller Area Network (CAN)

Modules Part II

Chapter 10: Flash Programming

Chapter 11: IQ – Math Library

Chapter 12: DSP/BIOS

Chapter 13: Boot – ROM

Chapter 14: FIR – Filter

Chapter 15: Digital Motor Control

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 TMS320F2812 belongs to a group of devices that are called "Digital Signal Controller (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 Processors (DSP)".

To begin with, let's introduce some definitions:

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

What is a Digital Signal Controller ?

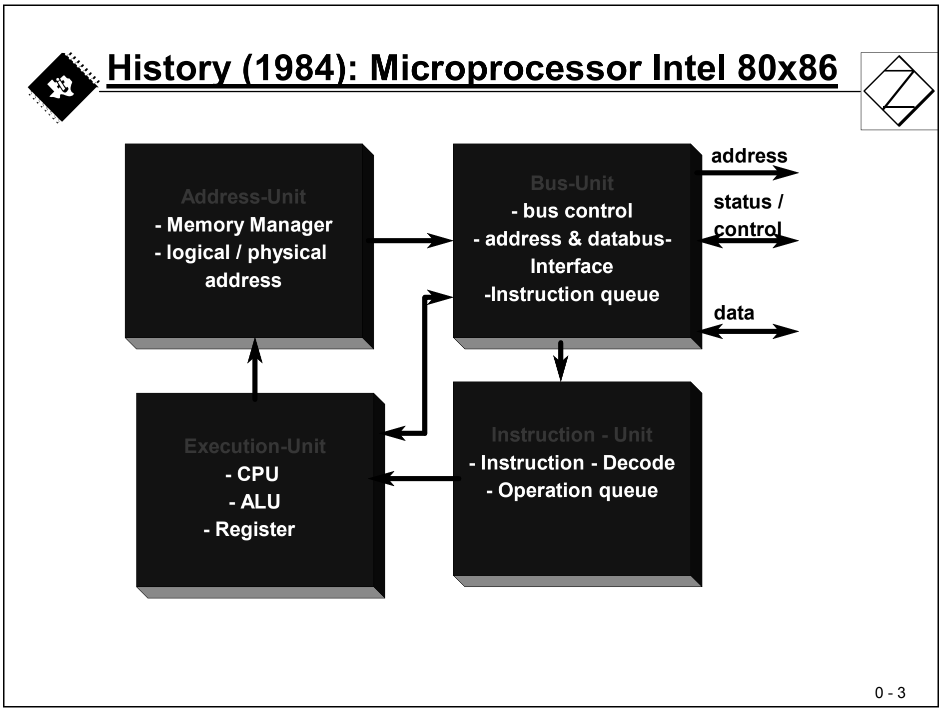
1. Microprocessor (µP):

- **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 to operate needs additional devices**

0 - 2

Microprocessors are based on a simple sequential procedural approach: Read next machine code instruction from code memory, decode instruction, read optional operands from data memory, execute instruction and 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.

The Intel 80x86: A typical Microprocessor



The Intel 8086 can be considered to be the veteran of all microprocessors. Inside this processor four units take care of 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 "cache".

We could go much deeper into the secrets 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's just keep in mind the basic operation of this type of device.

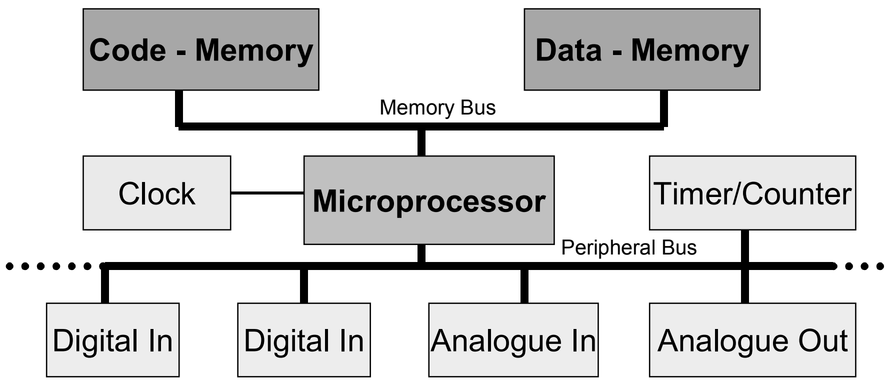
The Desktop – PC: a Micro Computer

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 analogue input/outputs.

Your Desktop – PC is a...

2. Micro Computer

- **Micro Computer = Microprocessor(µP) + Memory + Peripherals**
- **Example: your Desktop -PC**



0 - 4

Computer Peripherals

- **Peripherals include:**
  - **Digital Input / Output Lines**
  - **Analogue to Digital Converter (ADC)**
  - **Digital to Analogue Converter (DAC)**
  - **Timer / Counter units**
  - **Pulse Width Modulation (PWM) 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 …**

0 - 5

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 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 and – for (almost) nothing. This leads to a huge number of dedicated microcontroller families with totally different internal units, different instruction sets, different number of peripherals and internal memory spaces. No customer will ask for a microcontroller with an internal code memory size of 16Mbytes, if the application fits easily into 64Kbytes.

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 can't see them from outside the product. That is the reason why they are also called "embedded" computer or "embedded" controller. 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, the number giving the amount of bits of an operand that are processed 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.

System on Chip

3. Microcontroller (µC)

- **Nothing more than a Micro Computer 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**

0 - 6

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”.

Digital Signal Processor

4. Digital Signal Processor (DSP)

- **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**

0 - 7

What are the typical DSP algorithms?

- **The 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	$F(u)=\sum_{x=0}^{N-1} c(u).f(x).\cos\left[\frac{\pi}{2N}u(2x+1)\right]$

0 - 8

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's 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 that understands how a mobile phone or an autopilot of an airplane does work internally – a terrible thought.

To begin with, let's scale down the entire math's 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's (see Chapter 11).

Because of the TMS320F2812 is a fixed-point device, let's stay with this type of math's. If you look into chapter 1 of Texas Instruments C6000 Teaching CD-ROM, you will find a detailed discussion of pros and cons of fixed point versus floating point DSPs.

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 a common C-compiler system, e.g. Microsoft Visual Studio.Net**
- **A C-Code Solution could look like this:**

```
#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;
}
```

0 - 9

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 the 10 steps at one time. This will consume a considerable amount of 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 spend even more computing power.

6 Basic Operations of a SOP

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

- **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<3 , 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!

0 - 10

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)
```

0 - 11

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 slide 13 with slide 11 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;
}
```

0 - 12

DSP-Translation into machine code

Address	MCode	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 TMS320F2812
Space : 12 Code Memory ; 9 Data Memory
Execution Cycles : 10 @ 150MHz = 66 ns

0 - 13

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 first 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 TMS320F2812, a 32-bit fixed point Digital Signal Controller (DSC).

Digital Signal Controller (DSC)

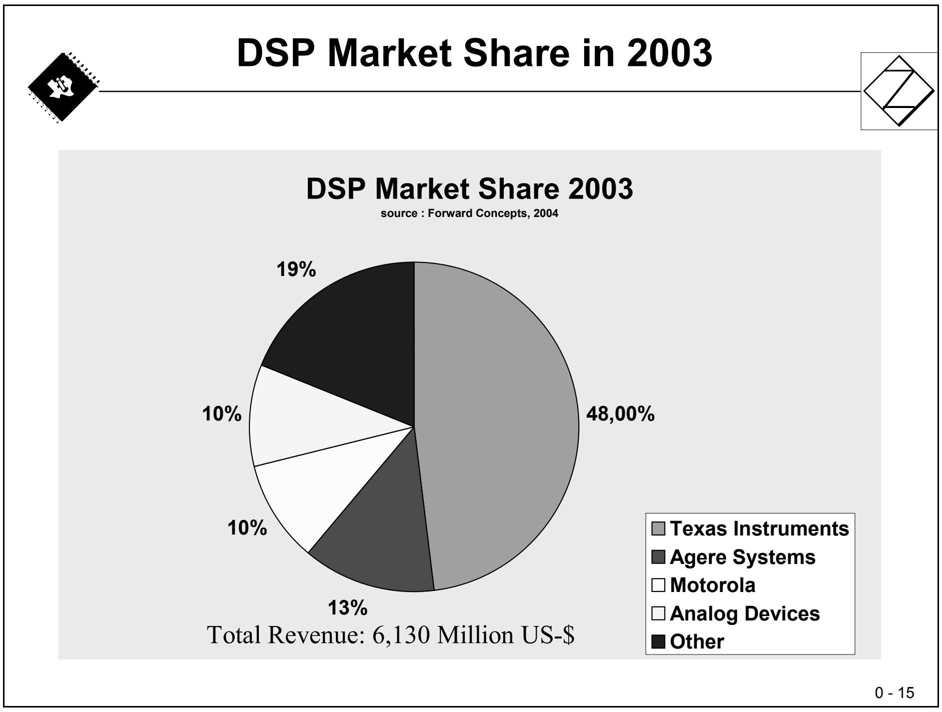
5. Digital Signal Controller (DSC)

- **recall: a Microcontroller(µC) 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 family.**

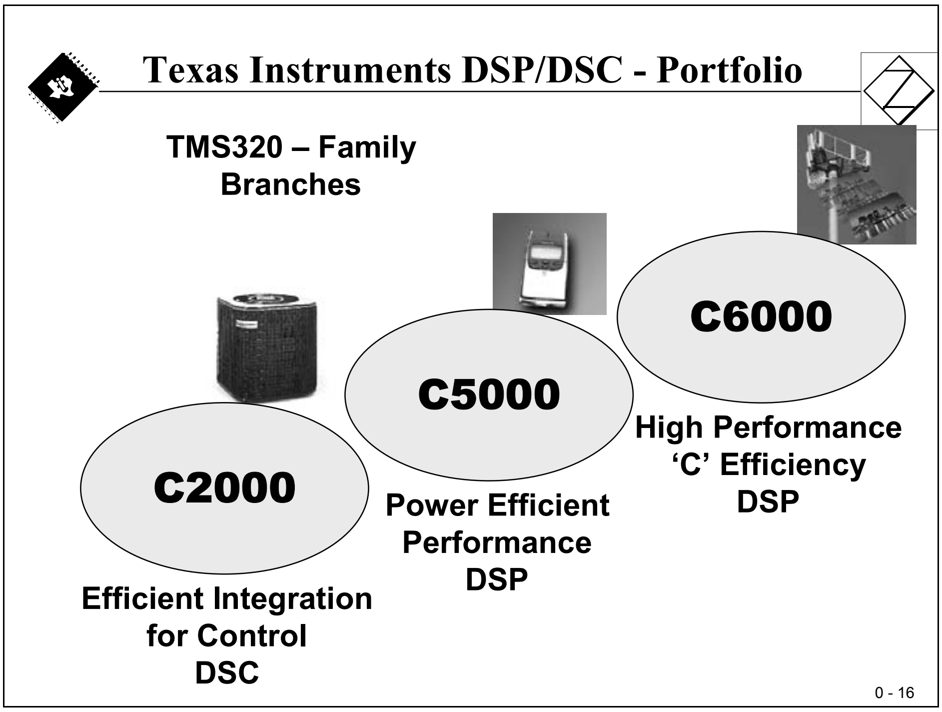
0 - 14

DSP Competition

There are only a few global players in the area of DSP and DSC. As you can see from the next slide (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.



Texas Instruments DSP – Portfolio



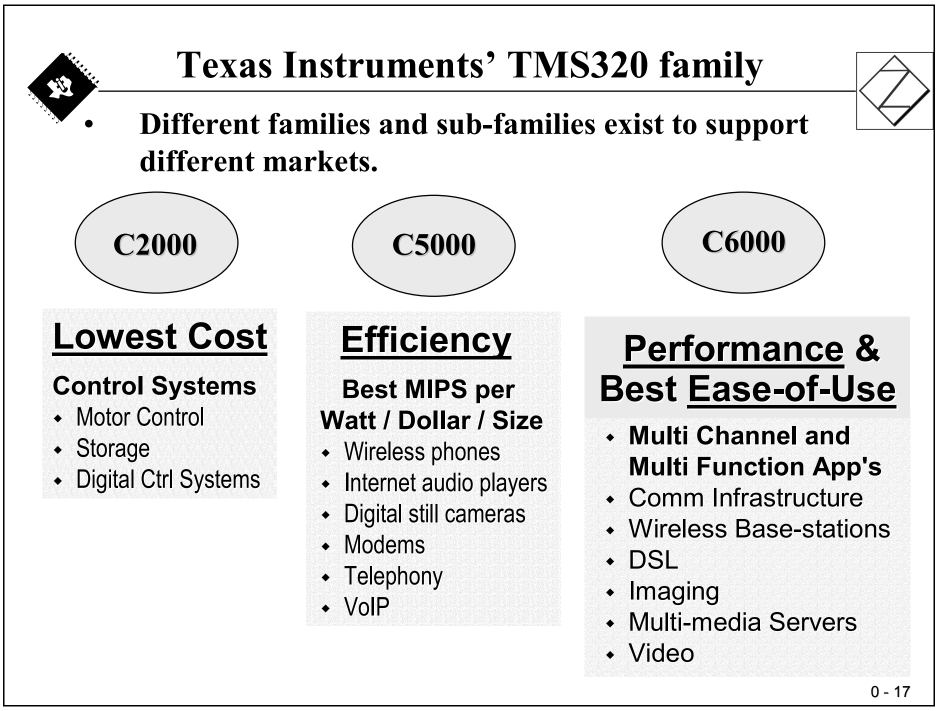
The DSP / DSC – portfolio of Texas instruments is split into three major device families, called C2000, C5000 and C6000.

The C6000 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.

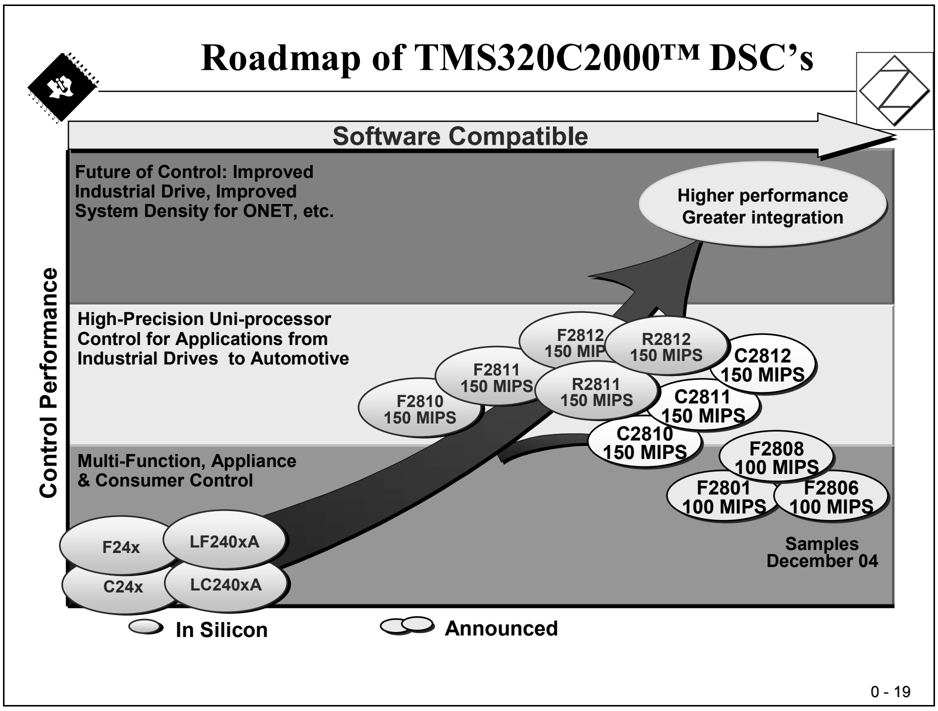
The C5000 family is focused on mobile systems with very efficient power consumption per MIPS. Its main application area is cell phone technology.

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.

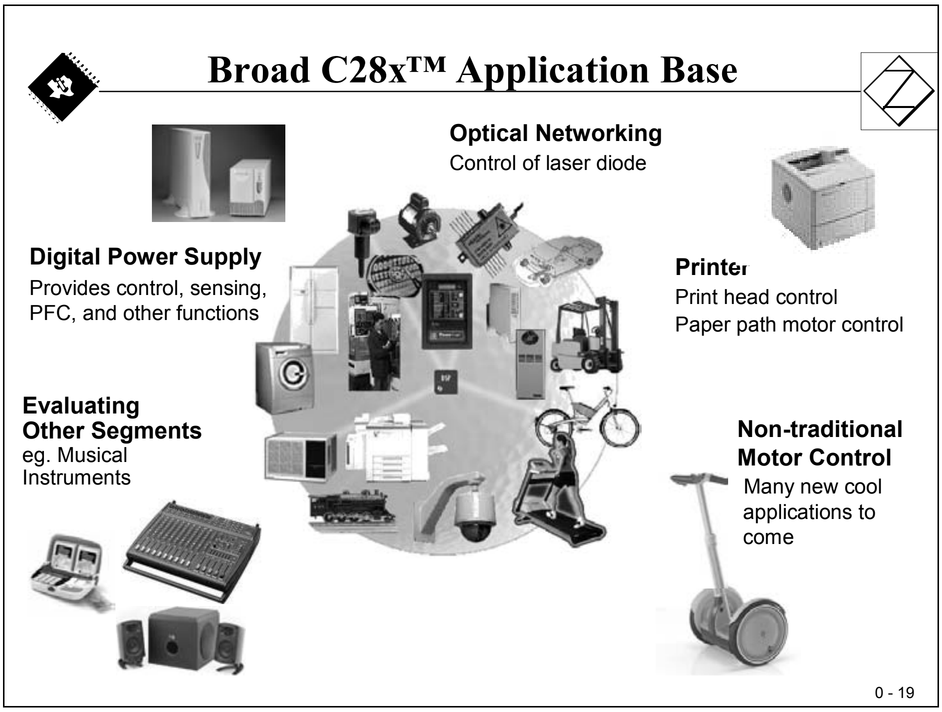
The next slide summarizes the main application areas for the 3 Texas Instruments families of DSP.



For the C2000 – branch we can distinguish between 2 groups of fixed-point DSC's: a 16-bit group, called TMS320C24x and a 32-bit group, called TMS320C28x.



TMS320F28x Roadmap



TI C2000: Portfolio for Embedded Applications

	F2812	F2810	LF2407A	LF2406A	LF2403A	LF2402A	LF2401A	LC2406A	LC2404A	LC2402A	LC2401A	F243	F241	F240	C242
CPU	32bit	32 bit	16 bit	16 bit	16 bit	16 bit	16 bit	16 bit	16 bit	16 bit	16 bit	16 bit	16 bit	16bit	16bit
MIPS	150	150	40	40	40	40	40	40	40	40	40	20	20	20	20
RAM (words)	18K	18K	2.5K	2.5K	1.0K	1.0K	1.0K	2.5K	1.5K	544	1.0K	544	544	544	544
ROM (words)								32K	16K	6K	8K				4K
Flash (words)	128K	64K	32K	32K	16K	8K	8K					8K	8K	16K	
BootROM (words)	4K	4K	256	256	256	256	256								
Event Manager CAP/QEP	6/6	6/6	6/4	6/4	3/2	3/2	1/0	6/4	6/4	3/2	1/0	3/2	3/2	4/2	3/2
PWM(CMP)	16	16	16	16	8	8	7	16	16	8	7	8	8	12	8
TIMER	7	7	4	4	2	2	2	4	4	2	2	2	2	3	2
ADC Resolution	12-bit	12-bit	10-bit	10-bit	10-bit	10-bit	10-bit	10 bit	10-bit	10-bit	10-bit	10-bit	10-bit	10-bit	10-bit
# ofChan	16	16	16	16	8	8	5	16	16	8	5	8	8	16	8
Conv time	200ns	200ns	500ns	500ns	500ns	500ns	500ns	375ns	375ns	425ns	500ns	900ns	900ns	6.1us	900ns
McBSP	✓	✓													
EXMIF	✓		✓									✓		✓	
Watch Dog	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
SPI	✓	✓	✓	✓	✓			✓	✓			✓	✓	✓	
SCI (UART)	2	2	1	1	1	1	1	1	1	1	1	1	1	1	1
CAN	✓	✓	✓	✓	✓			✓				✓			
Volts (V)	1.8 core 3.3 I/O	1.8core 3.3 I/O	3.3	3.3	3.3	3.3	3.3	3.3	3.3	3.3	3.3	5.0	5.0	5.0	5.0
# I/O	56	56	41	41	21	21	13	41	41	21	13	32	26	28	26
Package	176LQFP 179u*BGA	128LQFP	144LQPF	100LQPF	64LQFP	64PQFP	32LQFP	100LQFP	100LQFP	64PQFP	32LQFP	144LQFP	64PQFP 68PLCC	132PQFP	64PQFP 68PLCC

0 - 20