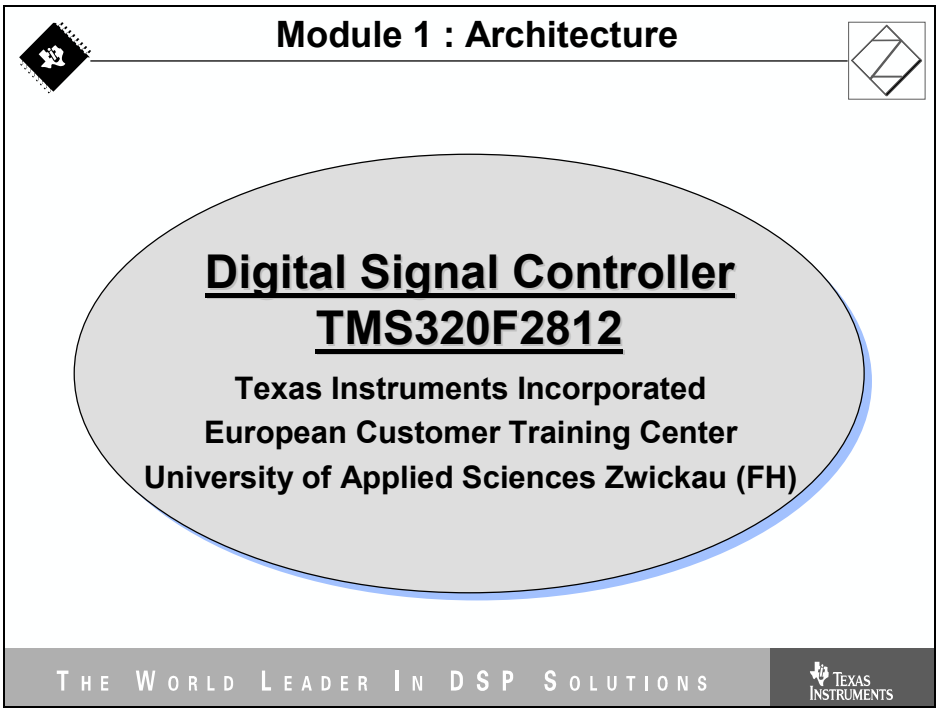
Architecture

Introduction

Since a Digital Signal Processor is capable of executing six basic operations in a single instruction cycle, the architecture of the TMS320F2812 must reflect this feature in some way. Recall this key point when we look into the details of this Digital Signal Controller (DSC). It will help you to understand the 'philosophy' behind the device with its different hardware units. Doing six basic math's operations is no magic; we will find all the hardware modules that are required to do so in this chapter.

Among other things, we will discuss the following parts of the architecture:

- Internal bus structure
- CPU
- Hardware Multiplier, Arithmetic-Logic-Unit, Hardware-Shifter
- Register Structure
- Memory Map

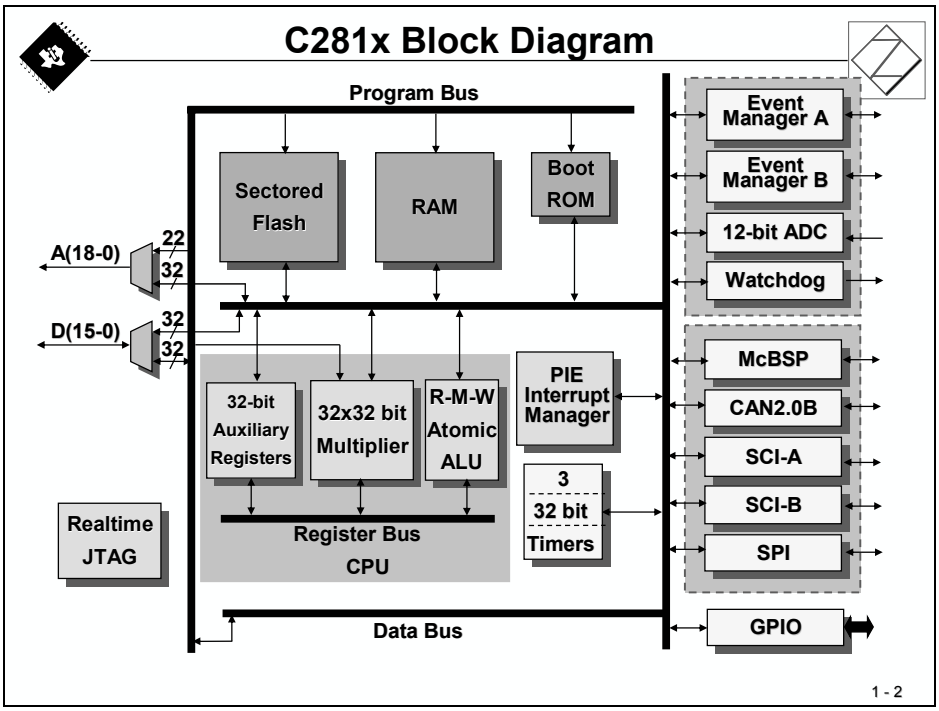


Module Topics

TMS320F2812 Block Diagram

The TMS320F2812 Block Diagram can be divided into 4 functional blocks:

- Internal & External Bus System
- Central Processing Unit (CPU)
- Memory
- Peripherals



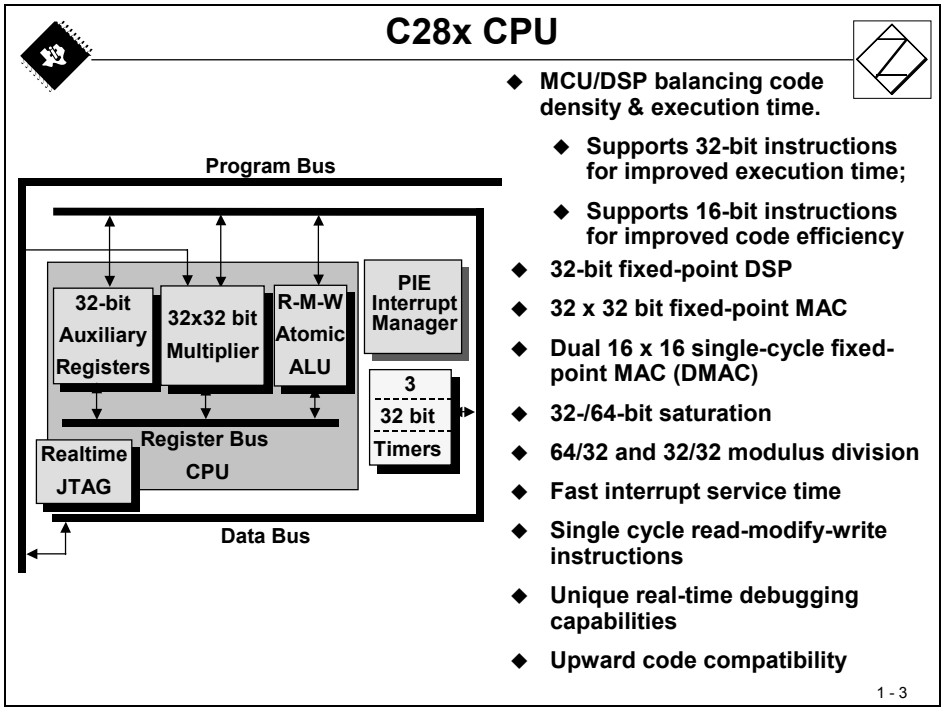
To be able to fetch two operands from memory into the central processing unit in a single clock cycle, the F2812 is equipped with two independent bus systems – Program Bus and Data Bus. This type of machine is called a "Harvard-Architecture". Due to the ability of the F2812 to read operands not only from data memory but also from program memory, Texas Instruments calls this device a "modified Harvard-Architecture". The "bypass"-arrow in the bottom left corner of slide 1-2 indicates this additional feature.

On the left side of the slide you will notice two multiplexer blocks for data (D15-D0) and address (A18-A0). It is an interface to connect external devices to the F2812. Please note that (1) the width of the external data bus is only 16 bits and that (2), you can't access the external program bus data and the data bus data at the same time. Compared to a single cycle for internal access to two 32-bit operands, it takes at least 4 cycles to do the same with external memory!

The F2812 CPU

The F2812 –CPU is able to execute most of the instructions to perform register-to-register operations and a range of instructions that are commonly used by micro controllers, e.g. byte packing and unpacking and bit manipulation in a single cycle. The architecture is also supported by powerful addressing modes, which allow the compiler as well as the assembly programmer to generate compact code that almost corresponds one-to-one with the C code.

The F2812 is as efficient in DSP math tasks as it is in the system control tasks that are typically handled by microcontroller devices. This efficiency removes the need for a second processor in many systems.



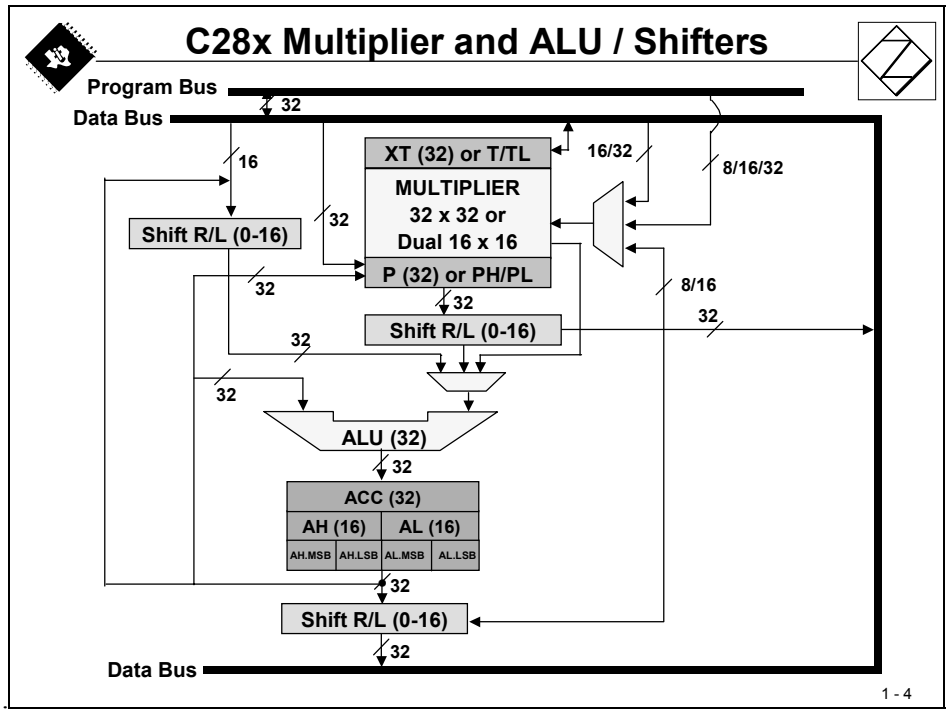
Three 32-bit timers can be used for general timing purposes or to generate hardware driven time periods for real time operating systems. The Peripheral Interrupt Expansion Manager (PIE) allows fast interrupt response to the various sources of external and internal signals and events. The PIE-Manager covers individual interrupt vectors for all sources.

A 32 by 32 bit hardware multiplier and a 32-bit arithmetic logic unit (ALU) can be used in parallel to execute a multiply and an add operation simultaneously. The auxiliary register bank is equipped with its own arithmetic logic unit (ARAU) – also used in parallel to perform pointer arithmetic.

The JTAG-interface is a very powerful tool to support real-time data exchange between the DSC and a host during the debug phase of project development. It is possible to watch variables while the code is running in real time, without any delay to the control code.

F2812 Math Units

The 32 x 32-bit Multiply and Accumulate (MAC) capabilities of the F2812 and its internal 64-bit processing capabilities, enable this DSC to efficiently handle higher numerical resolution problems that would otherwise demand a more expensive floating-point processor solution. Along with this is the capability to perform two 16 x 16-bit multiply and accumulate instructions simultaneously or Dual MAC's (DMAC).



Multiplication uses the XT register to hold the first operand and multiply it by a second operand that is loaded from memory. If XT is loaded from a data memory location and the second operand is fetched from a program memory location, a single-cycle multiply operation can be performed. The result of a multiplication is loaded into register P (product) or directly into the accumulator (ACC). Recall, if you multiply 32 x 32 bit numbers, what is the size of the result? Answer: 64-bit. The F2812 instruction set includes two groups of multiply operations to load both halves of the result into P and ACC.

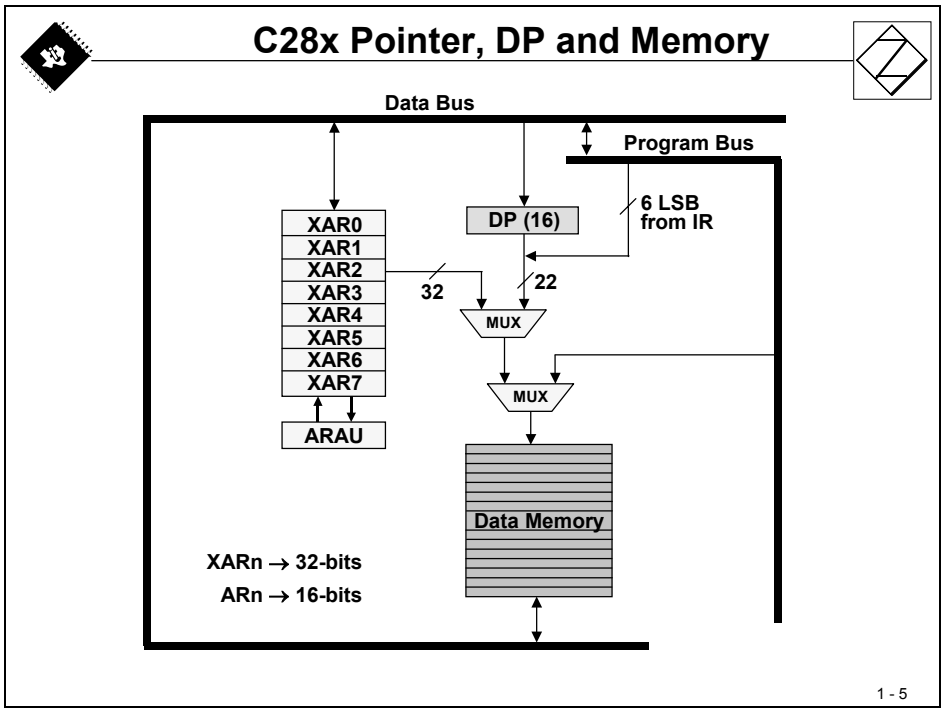
Three hardware shifters can be used in parallel to other hardware units of the CPU. Shifters are usually used to scale intermediate numbers in a real time control loop or to multiply/divide by 2^n.

The arithmetic logic unit (ALU) is doing the ‘rest’ of the math’s. The first operand is always the content of the Accumulator (ACC) or a part of it. The second operand for an operation is loaded from data memory, from program memory, from the P register or directly from the multiply unit.

Data Memory Access

Two basic methods are available to access data memory locations:

- Direct Addressing Mode
- Indirect Addressing Mode



Direct addressing mode generates the 22-bit address for a memory access from two sources – a 16-bit register "Data Page (DP)" for the highest 16 bits plus another 6 bits taken from the instruction. Advantage: Once DP is set, we can access any location of the selected page, in any order. Disadvantage: If the code needs to access another page, DP must be adjusted first.

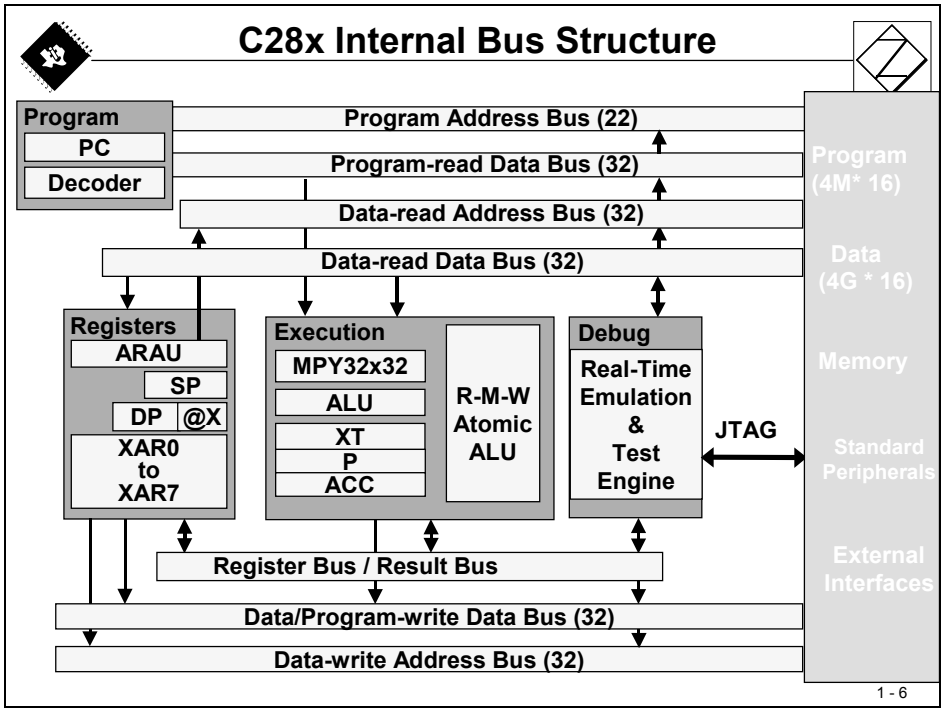
Indirect addressing mode uses one of eight 32-bit XARn registers to hold the 32-bit address of the operand. Advantage: With the help of the ARAU, pointer arithmetic is available in the same cycle in which an access to a data memory location is made. Disadvantage: A random access to data memory needs a new setup of the pointer register.

The auxiliary register arithmetic unit (ARAU) is able to perform pointer manipulations in the same clock cycle as access is made to a data memory location. The options for the ARAU are: post-increment, pre-decrement, index addition and subtraction, stack relative operation, circular addressing and bit-reverse addressing with additional options.

Internal Bus Structure

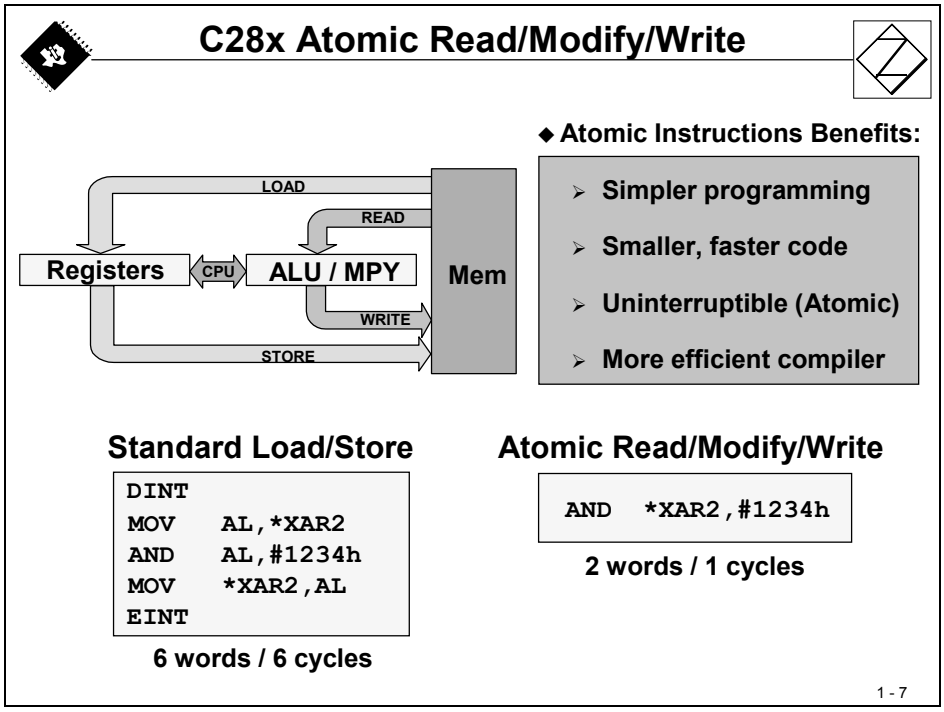
As with many DSP type devices, multiple busses are used to move data between memory locations, peripheral units and the CPU. The F2812 memory bus architecture contains:

- A program read bus (22 bit address line and 32 bit data line)
- A data read bus (32 bit address line and 32 bit data line)
- A data write bus (32 bit address line and 32 bit data line)



The 32-bit-wide data busses enable single cycle 32-bit operations. This multiple bus architecture, known as a Harvard Bus Architecture enables the C28x to fetch an instruction, read a data value and write a data value in a single cycle. All peripherals and memories are attached to the memory bus and will prioritise memory accesses.

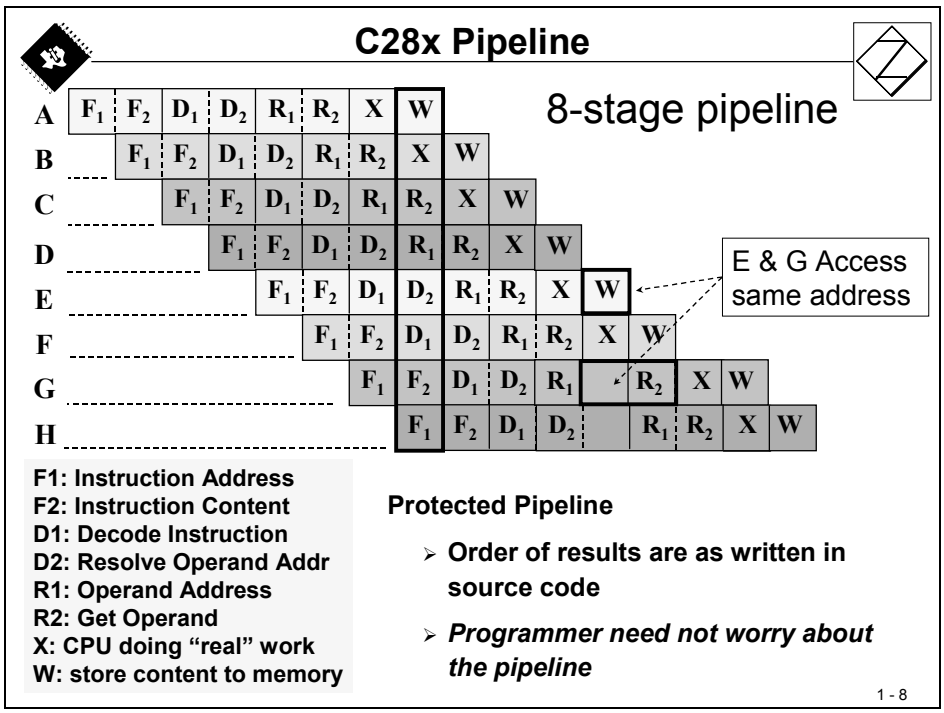
Atomic Arithmetic Logic Unit (ALU)



Atomics are small common instructions that are non-interruptible. The atomic ALU capability supports instructions and code that manages tasks and processes. These instructions usually execute several cycles faster than traditional coding.

Instruction Pipeline

The F2812 uses a special 8-stage protected pipeline to maximize the throughput. This protected pipeline prevents a write to and a read from the same location from occurring out of sequence. This pipelining also enables the C28x to execute at high speeds without resorting to expensive high-speed memories. Special branch-look-ahead hardware minimizes the delay when jumping to another address. Special conditional store operations further improve the system performance.



Each instruction goes through 8 stages until final completion. Once the pipeline is filled with instructions, one instruction is executed per clock cycle. For a 150MHz device, this equates to 6.67ns per instruction. The stages are:

F1: Generate Instruction Address at program bus address lines.

F2: Read the instruction from program bus data lines.

D1: Decode Instruction

D2: Calculate Address information for operand(s) of the instruction

R1: Load operand(s) address to data and/or program bus address lines

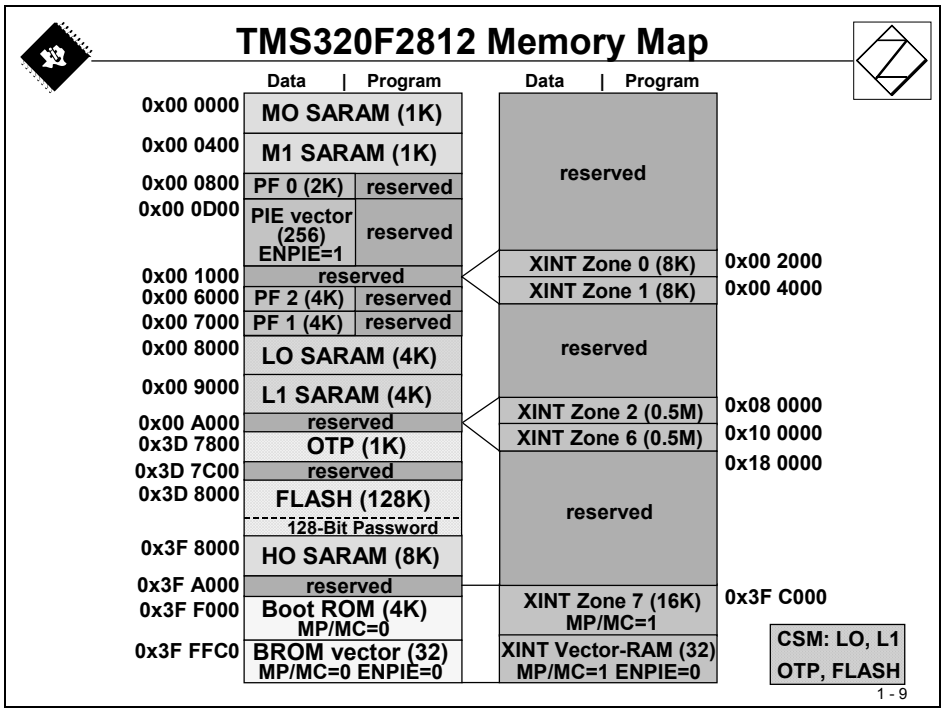
R2: Read Operand

X: Execute the instruction

W: Write back result to data memory

Memory Map

The memory space on the F2812 is divided into program and data space. There are several different types of memory available that can be used as both program or data space. They include flash memory, single access RAM (SARAM), expanded SARAM, and Boot ROM which is factory programmed with boot software routines or standard tables used in math related algorithms. Memory space width is always 16 bit.



The F2812 can access memory both on and off the chip. The F2812 uses 32-bit data addresses and 22-bit program addresses. This allows for a total address reach of 4G words (1 word = 16 bits) in data space and 4M words in program space. Memory blocks on all F2812 designs are uniformly mapped to both program and data space.

The memory map above shows the different blocks of memory available to the program and data space.

The non-volatile internal memory consists of a group of FLASH-memory sections, a boot-ROM for up to six reset-startup options and a one-time-programmable (OTP) area. FLASH and OTP are usually used to store control code for the application and/or data that must be present at reset. To load information into FLASH and OTP one need to use a dedicated download program, that is also part of the Texas Instruments Code Composer Studio integrated design environment.

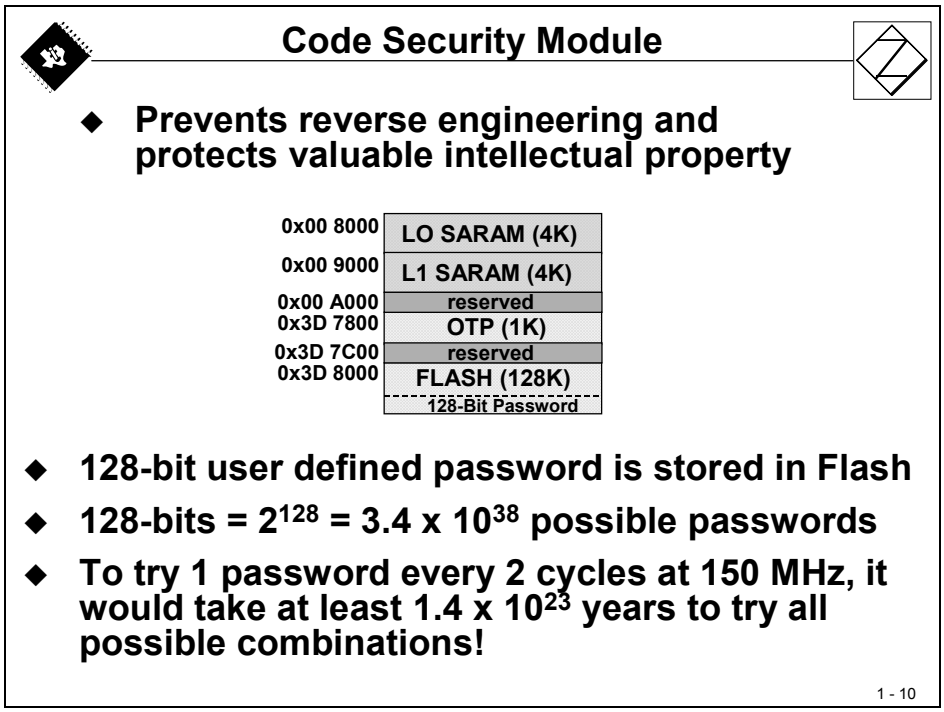
Volatile Memory is split into 5 areas (M0, M1, L0, L1 and H0) that can be used both as code memory and data memory.

PF0, PF1 and PF2 are Peripheral Frames that cover control and status registers of all peripheral units (“Memory Mapped Registers”).

Code Security Module

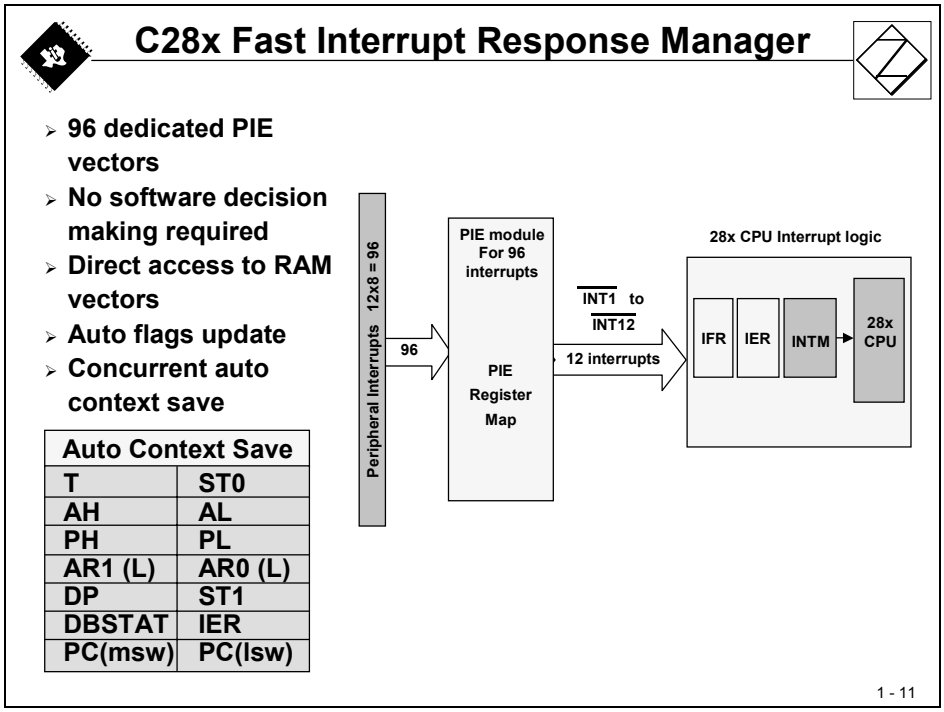
There is an internal security module available in all F28x family members. It is based on a 128-bit password that is written by the ***software developer*** into the last 8 memory spaces of the internal FLASH (0x3F 7FF8 to 0x3F 7FFF). Once a pattern is written into this area, all further accesses to any of the memory areas covered by this Code Security Module (CSM) are denied, as long as the ***user*** does not write an identical pattern into password registers of frame PF0.

NOTE: If you write any pattern into the password area ***by accident***, there is no way to get access to this device any more! So please be careful and do not upset your laboratory technician!



Interrupt Response

The fast interrupt response, with automatic “context” save of critical registers, resulting in a device that is capable of servicing many asynchronous events with minimal latency. Here “context” means all the registers you need to save so that you can go away and carry out some other process, then come back to exactly where you left. F2812 devices implement a zero cycle penalty to save and restore the 14 registers during an interrupt. This feature helps to reduce the interrupt service routine overheads.

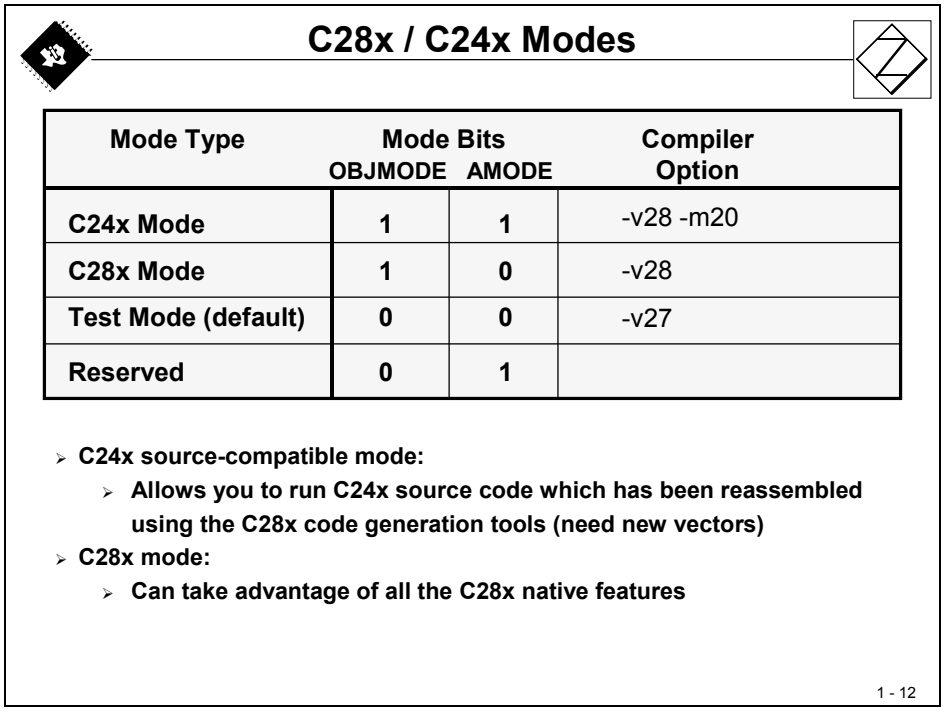


We will look in detail into the F2812’s interrupt system in Module 4 of this tutorial. The Peripheral Interrupt Expansion (PIE) – Unit allows the user to specify individual interrupt service routines for up to 96 internal and external interrupt events. All possible 96 interrupt sources share 14 maskable interrupt lines (INT1 to INT14), 12 of them are controlled by the PIE – module.

The auto context save loads 14 important CPU registers, shown at the slide above, into a stack memory, which is pointed to by a stack pointer (SP) register. The stack is part of the data memory and must reside in the lower 64K words of data memory.

Operating Modes

The F2812 is one of several members of the fixed-point generations of digital signal processors (DSP's) in the TMS320 family. The F2812 is source-code and object-code compatible with the C27x. In addition, the F2812 is source code compatible with the 24x/240x DSP and previously written code can be reassembled to run on a F2812 device. This allows for migration of existing code onto the F2812.



Actually the F28x silicon is able to operate in three different modes:

- C28x – Mode - takes advantage of all 32-bit features of the device
- C24x – Mode - source code compatibility to the 16-bit family members
- C27x – Mode - intermediate operating mode, test purposes only.

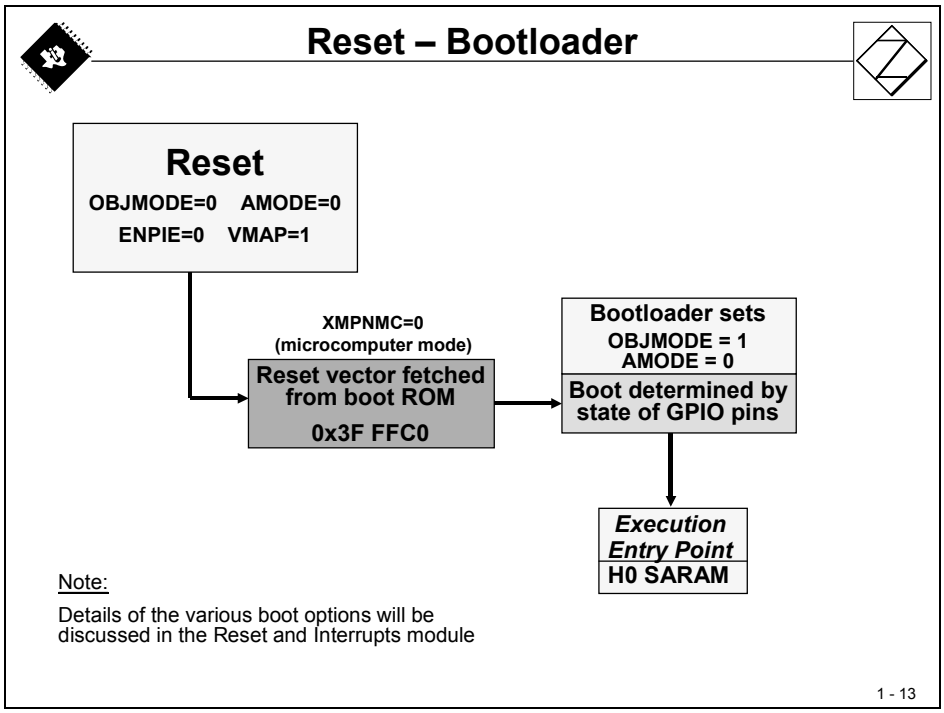
After RESET, the device behaves like a C27x device. To take advantage of the full computing power of a C28x device, the control flag "OBJMODE" must be set to 1. If you are using a C-compiler generated program, the start function of the C environment takes care of setting OBJMODE to 1. But, if you decide to develop an assembler language based solution, your first task after reset is to bring the device into C28x – mode manually.

Reset Behaviour

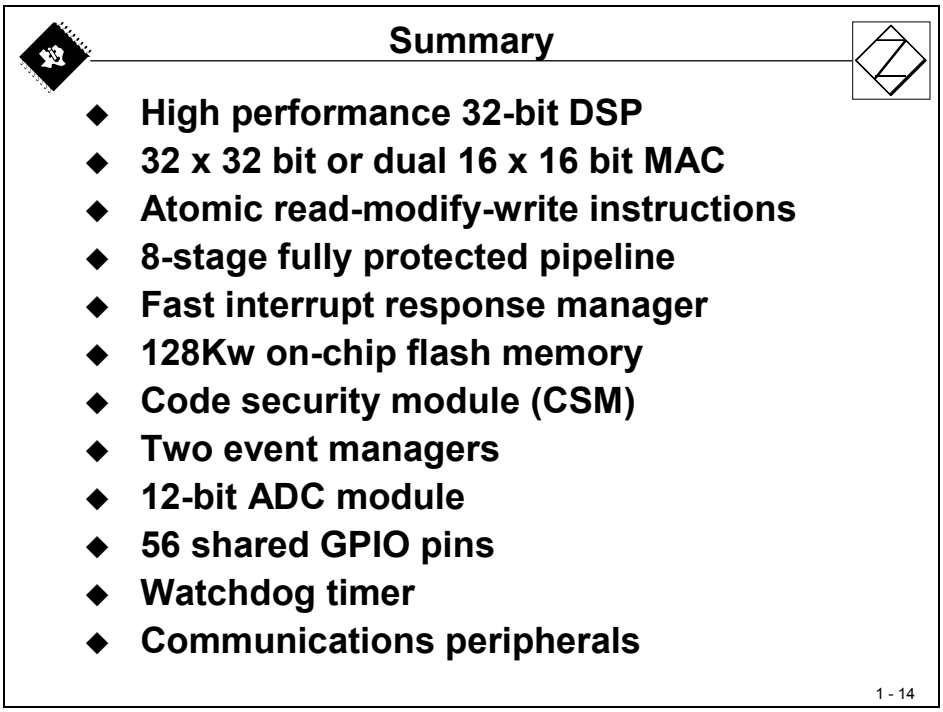
After a valid RESET-signal is applied to the F2812, the following sequence depends on some external pins on this DSC.

If the pin “XMPNMC” is connected to ‘1’, the F2812 tries to load the next instruction from address 0x3F FFC0 from an ***external memory*** at this position. This is the “Microprocessor”-Mode, loading instructions from external code memory.

If the pin “XMPNMC” is connected to ‘0’, the F2812 jumps directly into the ***internal boot code*** memory at address 0x3F FFC0. We call this mode “Microcontroller”-Mode. The code in this memory location has been developed by TI to be able to distinguish between 6 different start options for the F2812. The actual option is selected by the status of 4 more pins during this phase. For our tutorial we use the volatile memory H0 as code memory and its first address as the execution entry point.



Summary of TMS320F2812 Architecture



This page was intentionally left blank.