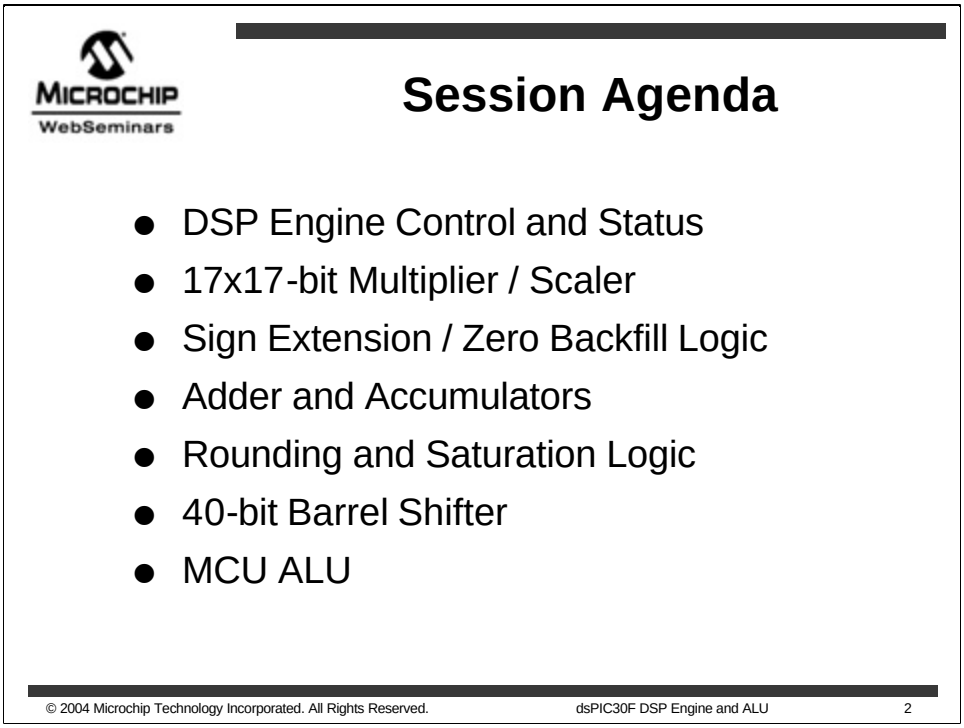
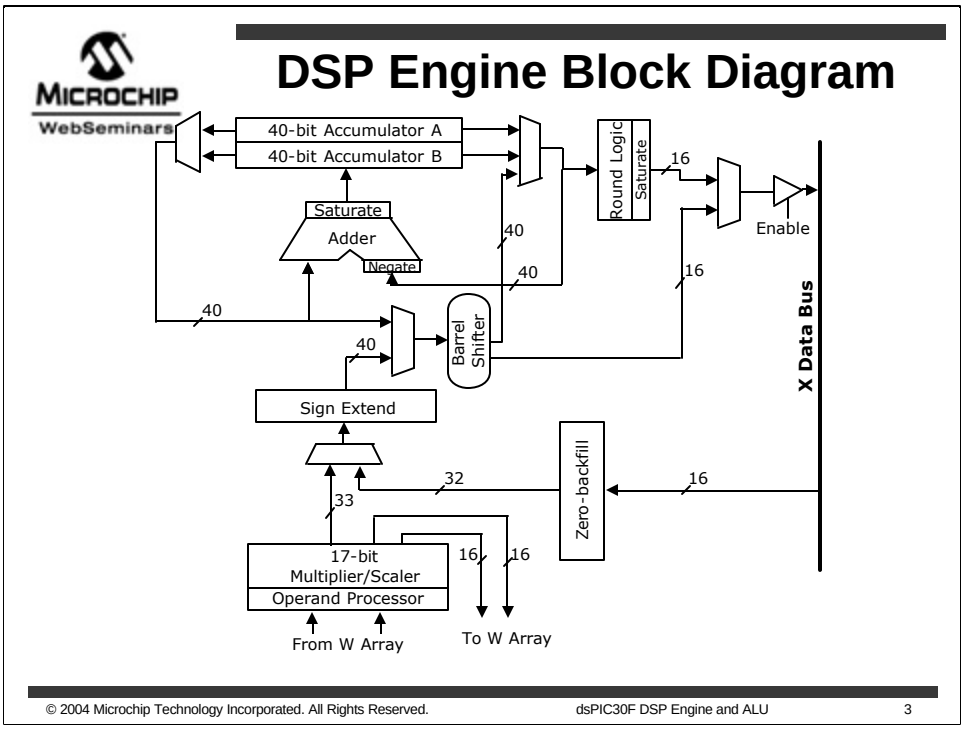
Welcome to the dsPIC30F DSP Engine and ALU Web seminar. My name is Ralph Fulchiero and in this session you're going to learn about the internal operation of the DSP Engine and ALU. The information in this seminar will allow you to take advantage of the most powerful features of the dsPIC30F architecture.



This is the agenda for this presentation. We will start by taking a look at the various special function registers used to control the operation of the Central Processing Unit, or CPU. We will then study the multiplier and other associated hardware functionality such as scaling, sign-extension and zero-backfill. Subsequently, we will learn about the adder and the two DSP accumulators, data rounding and saturation, and the Barrel Shifter. Last but not the least, we will briefly survey the features of the MCU Arithmetic and Logic Unit, or ALU.



Shown here is a high-level block diagram of the dsPIC30F DSP Engine. Gaining an understanding of the DSP Engine will enable you to take advantage of the dsPIC30F architecture and write more efficient software.

This block diagram shows the primary components of the DSP Engine, along with the data paths between the components, the Working Register array and the X Data Bus.

During this presentation, we will discuss each component of the DSP Engine, namely:

17-by-17 bit Multiplier and Scaler,

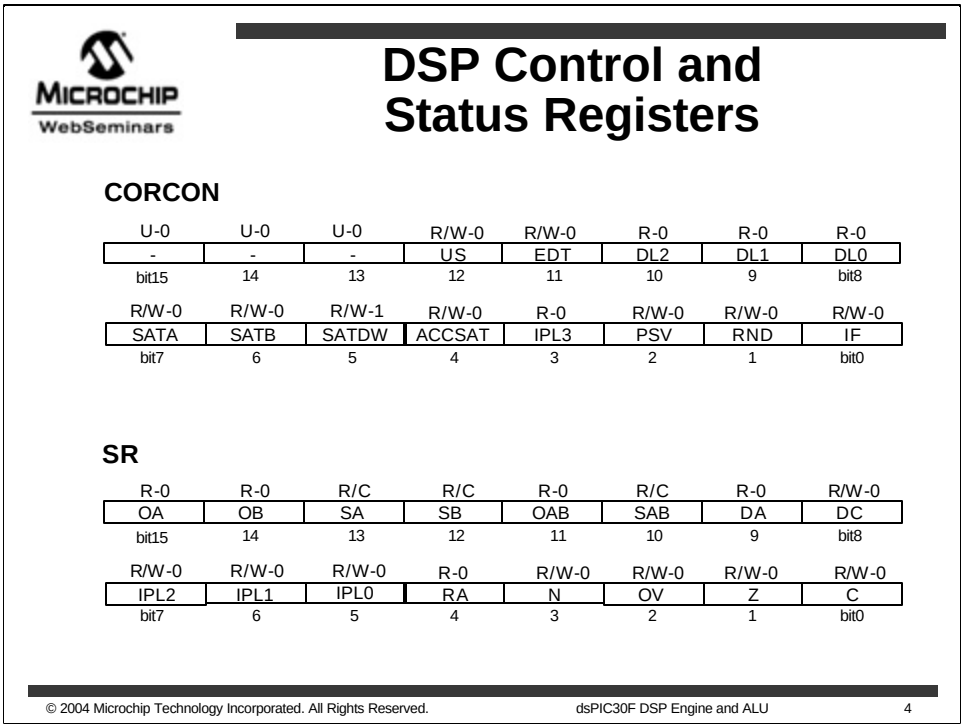
Sign-Extend and Zero-Backfill Logic,

40-bit Adder with Negate and Saturation logic,

Two 40-bit Accumulators,

40-bit Barrel Shifter, and

Round and Saturation Logic

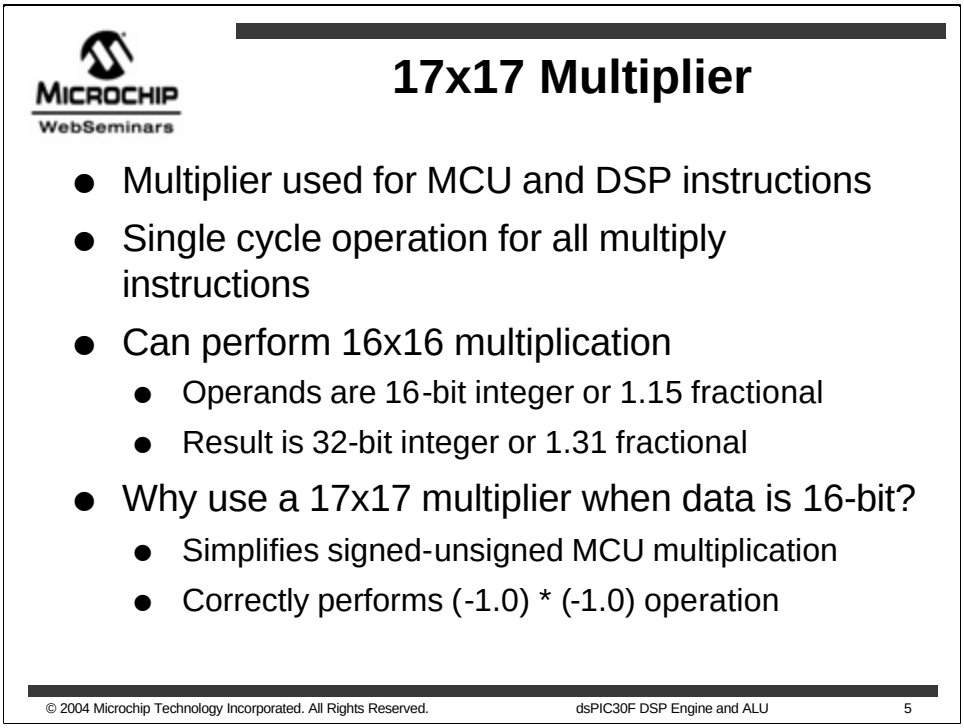


This foil shows the individual bits of the CORCON register *(pronounced "core-con")* . This register controls many important features of the dsPIC30F, but for this presentation, we are only going to focus on the DSP Engine control bits, which are highlighted.

The US *(pronounced "U-S")* bit selects between signed and unsigned DSP multiplication operations. The SATA *(pronounced "sat-a")* bit enables saturation for Accumulator A, while the SATB *(pronounced "sat-b")* bit enables saturation for Accumulator B. The mode of saturation, either 9.31 or 1.31 *(pronounced "nine dot thirty-one or one dot thirty-one")*, is controlled by the ACCSAT *(pronounced "ack-sat")* bit. The SATDW *(pronounced "sat-d-w")* bit enables data space write saturation, which controls how the accumulators are written to memory. The RND *(pronounced "R-N-D")* bit selects the rounding mode, either convergent or conventional, and the IF *(pronounced "I-F")* bit selects between fractional and integer DSP multiplication operations.

The Status Register, or SR, includes various flags that indicate the current state of the DSP Engine. The SA and SB bits indicate if saturation occurred in accumulator A and B respectively. Similarly the OA and OB bits indicate if overflow occurred.

The OAB bit is simply the logical "OR" of the OA and OB bits. Similarly, the SAB bit is a logical "OR" of the respective SA and SB bits. The OAB and SAB bits provide a convenient means of examining DSP status. The SAB bit is unique, in that writing a "0" to it will clear both SA and SB. The SA, SB and SAB bits are described as being "sticky". That is, once they are set, they must be manually cleared by the user. This is a nice feature, because if one performs a series of operations, the status register may be checked just once at the end of the computation to determine if saturation occurred.

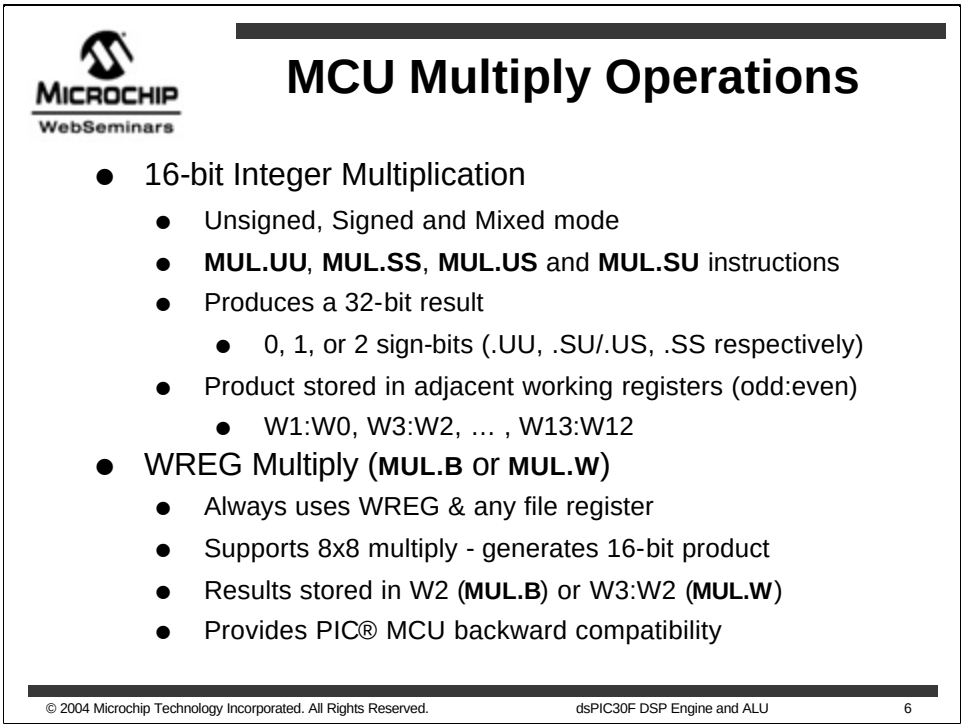


The dsPIC30F provides a 17-bit-by-17-bit multiplier which executes all multiplication operations in one instruction cycle. The Multiplier is a common functional block found on other DSP products but not in most MCUs. On the dsPIC30F, this multiplier is shared between the DSP and MCU instructions. The multiplier can perform 16-by-16 bit multiplication.

The operands may either be signed or unsigned 16-bit integer or 1.15 fractional operands. 1.15, also known as Q15, is the default fractional data representation used by DSP operations on the dsPIC30F, and contains 1 sign bit and 15 fractional bits.

The results of the multiplication can be either a 32-bit fractional number in 1.31 format, or a 32-bit signed or unsigned integer. The output of the multiplier and scaler is fed into a sign-extension block, which produces a 40-bit sign-extended result.

Even though both inputs to the multiplier are 16-bit data, a 17-bit multiplier is used in order to correctly handle the fixed-point computation of (-1.0 * -1.0) *(pronounced "minus one point zero" times "minus one point zero")* or (0x8000 * 0x8000) *(pronounced "hex eight thousand" times "hex eight thousand")*. Before the multiplication is performed, both 16-bit operands are sign-extended to 17-bits.



The MCU Multiply instructions provide integer multiplication support for unsigned, signed or mixed-sign multiply operations. The instruction extension determines the type of multiplication being performed:

The ".UU" extension is for multiplying an unsigned number with an unsigned number

The ".SS" extension is for a signed-signed multiplication

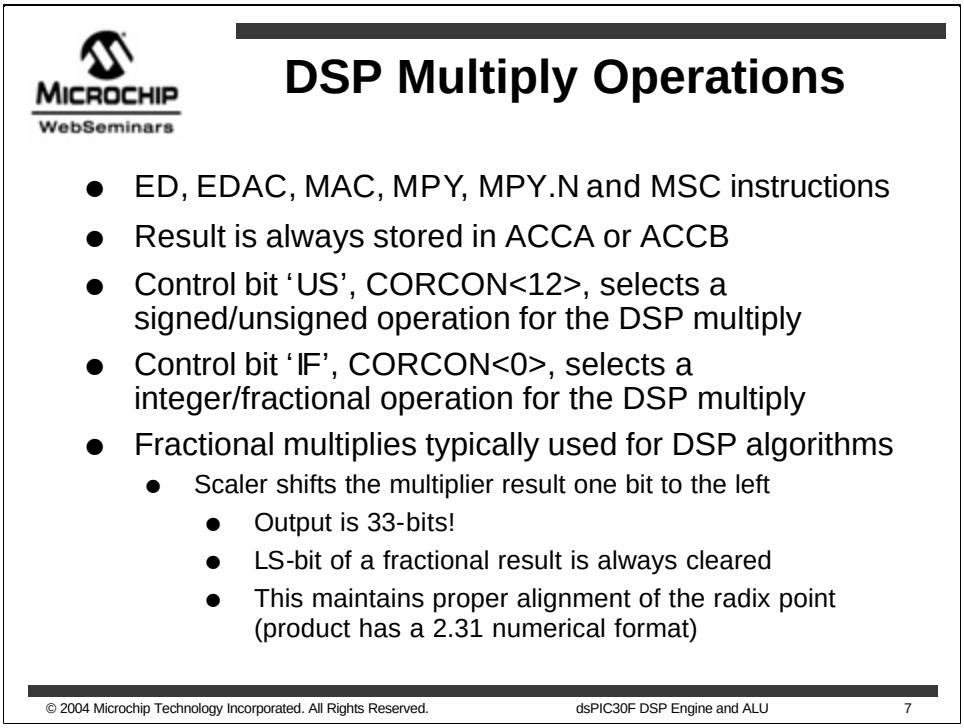
The ".US" extension is for an unsigned-signed multiplication

The ".SU" extension is for a signed-unsigned multiplication

These instructions perform a 16-bit-by-16-bit integer multiply operation, and generate a 32-bit result which is stored in any 2 adjacent working registers. The WREG multiply instruction operates on the WREG *(pronounced "W-Redge"),* which is defined to be working register W0, and any file register in the near data region. This instruction supports a byte and a word mode multiply:

The MUL.B instruction is a 8-bit x 8-bit multiply, with the results stored in W2, whereas the MUL.W instruction is a 16-bit x 16-bit multiply, with the results stored in W3 and W2.

The MUL instruction provides backward compatibility for the PIC® MCU "MULWF" *(pronounced "MUL-W-F")* instruction.

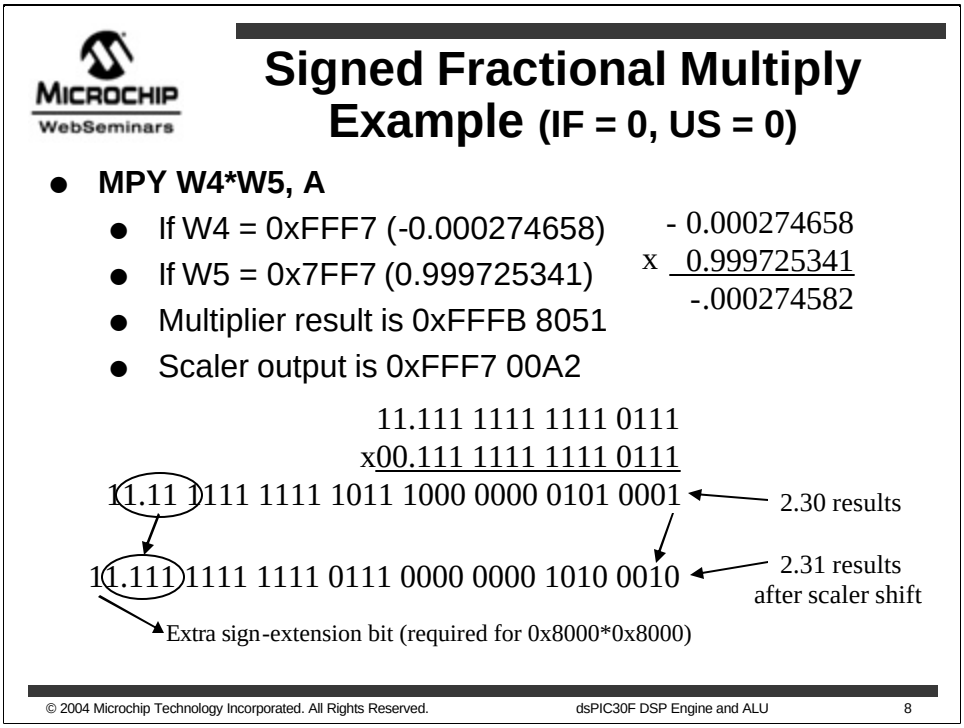


When a DSP multiplication is performed, the results are stored in one of the 40-bit Accumulators, ACCA or ACCB *(pronounced "ack-a or ack-b")*. The accumulator is specified as one of the operands in the instruction.

The "IF" control bit in the CORCON register determines how the DSP multiplication operation will be performed. If IF equals 0, then a fractional multiplication is performed. This means that after the multiplication is done, the scaler will shift the result one bit to the left, generating an output with 2 sign-bits and 31 fractional bits, which is called the 2.31 format *(pronounced " two dot thirty-one format")*. This shift is necessary to maintain proper alignment of the radix point, and always clears the Least Significant Bit, or LSB, of the result.

Conversely, if IF equals 1, then an integer multiplication is performed. When an integer DSP multiplication is performed, the result is not affected by the scaler and passes out of the multiplier as a 32-bit number.

The "US" control bit in the CORCON register determines if the data operands should be treated as signed numbers or unsigned numbers. When operands are treated as unsigned numbers, the sign-extension bit, which is the 17th bit of the 17 by17 multiplier, will automatically be loaded with "0". Setting the US bit to "0" and the IF bit to "1" puts the multiplier in an unsigned-integer mode, which is particularly useful for efficient execution of cryptography algorithms.



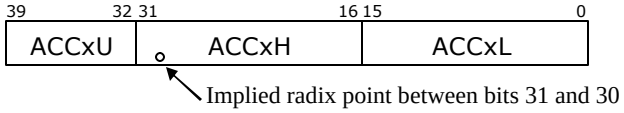
This example shows how a DSP multiplication is performed on the dsPIC30F. In this example, the contents of W4 are multiplied with the contents of W5, and the result is stored in Accumulator A. Since IF equals 0 in this example, a fractional multiply is performed.

Since this is a fractional multiply, the scaler shifts the integer result left one bit, creating the multiplier output which consists of 33-bits. The 33-bit result consists of 2 sign-bits and 31 fractional bits. The user will never see this 33-bit intermediate result, because the sign-extension logic will convert this to a 40-bit number before it is loaded to Accumulator A.

Even though the result of a fractional multiply has 31 fractional bits, the LSB is always “0” due to the scaler shift. This means that resolution of the fractional multiplier is “2 raised to -30”, and not “2 raised to -31”.

Zero Backfill & Sign Extension Control Logic

- Control logic used to interface 16-bit and 32-bit data with the 40-bit accumulator
  - Think of it as a “data formatter” which allows data to be properly placed in the 40-bit accumulator
- Sign extension logic sign extends from bit 32 or 31 through all of ACCxU
  - ACCxU = 0xFF for negative values
  - ACCxU = 0x00 for positive values
- Radix point resides between bits 31 and 30



The Zero Backfill and Sign Extension logic is used to interface the X Data Bus and the output of the multiplier to the 40-bit Accumulators, ACCA and ACCB.

Each accumulator consists of 3 adjacent registers:

Accumulator Upper, which is the upper byte, bits 39 through 32,

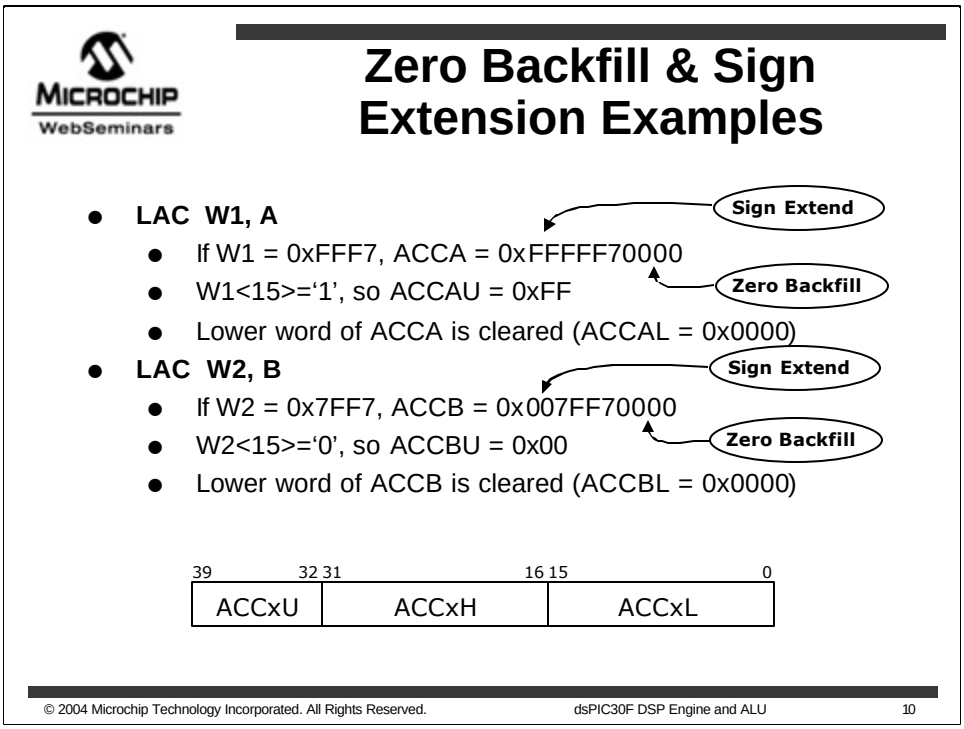
Accumulator High, which is the middle word, bits 31 through 16, and

Accumulator Lower, which is the lower word, bits 15 through 0.

The implicit radix point for fractional numbers is set between bits 31 and 30.

When an accumulator is loaded using the Load Accumulator, or LAC, instruction, Accumulator Lower will be set to zero from the zero backfill logic, and all bits in Accumulator Upper will take the value of accumulator bit 31 from the sign extension logic.

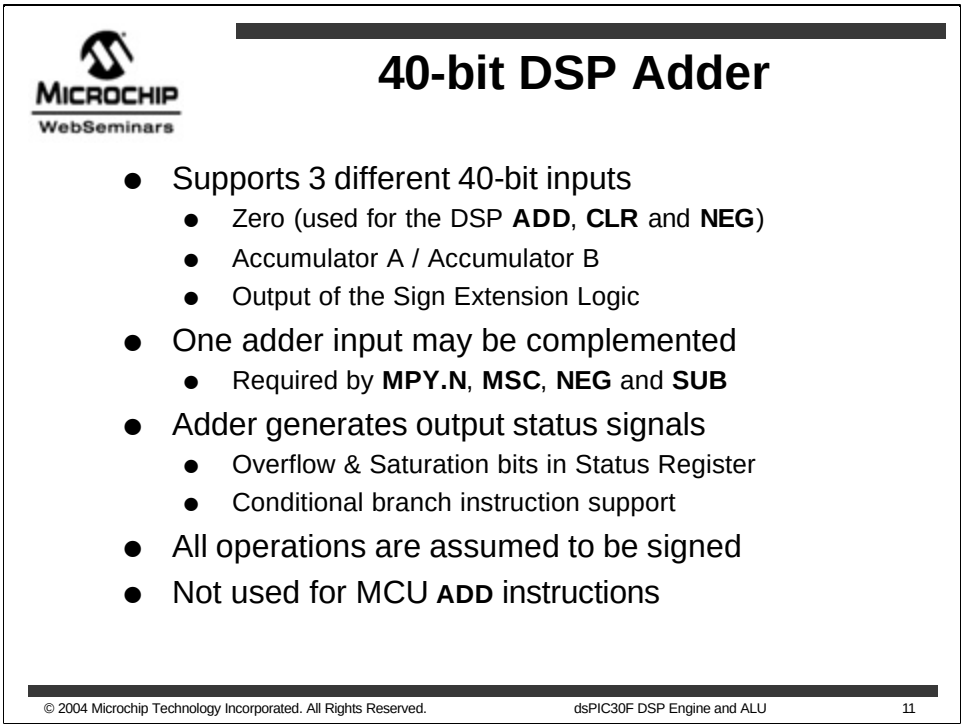
For DSP multiply instructions, the 33-bit result of the Scaler serves as the input to the Sign Extend logic, which will sign extend the result to 40 bits.



These 3 examples demonstrate how the Zero Backfill and Sign Extension logic operates. The effects of the Zero Backfill logic are shown in green, and the effects of the Sign Extension logic are shown in light-blue.

In the first example, the contents of W1 are loaded into Accumulator A. Since W1 is negative (as bit 15 is 1), the Sign Extension logic sets Accumulator A Upper to 0xFF *(pronounced as "hex F-F")*. The Zero Backfill logic sets Accumulator A Lower to zero for all LAC instructions.

In the second example, the contents of W2 are loaded into Accumulator B. Since W2 is positive (as bit 15 is 0), the Sign Extension logic sets Accumulator A Upper to zero. The Zero Backfill logic sets Accumulator A Lower to zero for all LAC instructions.

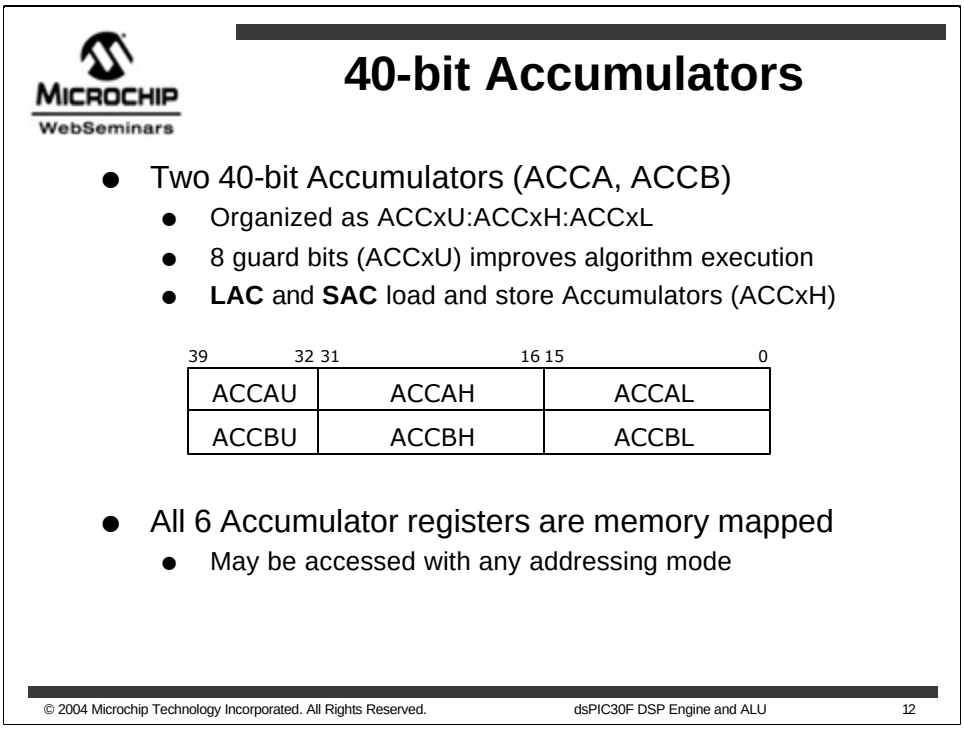


The 40-bit DSP Adder supports three different inputs. The inputs may be 0, or come from an Accumulator or the Sign Extension logic. The selection of the inputs is based upon the instruction executed, and is transparent to the user, meaning that it is handled by the instruction decoding.

All Adder operations are signed, and one input to the Adder may also be negated. This feature provides support for the DSP instructions Multiply and Negate, Multiply and Subtract, Accumulator Negate and Accumulator Subtract.

The Adder generates status bits which indicate overflow and saturation. The different saturation modes are user selectable, and are discussed later in this session. The architecture does provide conditional branch instruction support for these status bits, which facilitates efficient handling of overflow and saturation conditions in software.

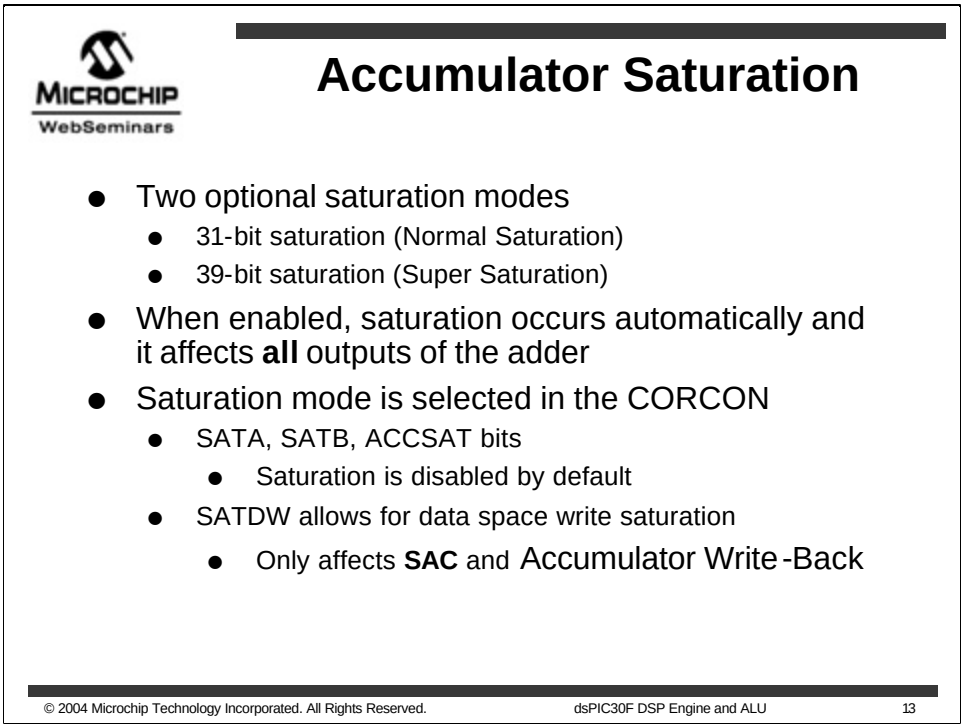
Lastly, since there is no data path between the working register array and the 40-bit Adder, MCU ADD instructions do not utilize the DSP Adder. They use the MCU ALU, which will be discussed later in this presentation.



Each of the two 40-bit accumulators, A and B, is composed of an upper byte and two 16-bit words arranged as a high word and a low word. One can think of the upper byte, Accumulator Upper, as providing an extra 8 guard bits which can be useful when executing a long series of operations on the accumulator.

When the accumulator is loaded by executing a LAC instruction, or as the result of a DSP multiply instruction, the Accumulator Upper register contains sign extension bits. However, as one performs operations on the contents of the accumulator, it is quite possible that the result of the operation may “grow” or overflow into the upper guard bits of the accumulator, provided, of course, that the selected saturation mode allows for this. When this occurs, the guard bits provide computational headroom for accurately operating with these larger numbers.

Since the accumulators are memory mapped, each register may alternatively be accessed as a general data memory location using MCU instructions.



The dsPIC30F supports two optional saturation modes which always affect the output of the 40-bit Adder: 31-bit Saturation (referred to as Normal Saturation) and 39-bit Saturation (referred to as Super Saturation). The effects of these saturation modes will be seen in the next few slides.

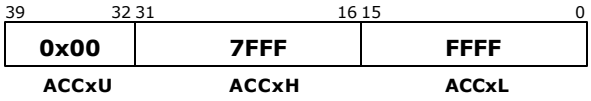
Selecting the saturation mode is controlled by several bits in the CORCON register. The SATA bit enables or disables saturation for Accumulator A, and the SATB bit enables or disables saturation for Accumulator B.

When saturation is enabled, the ACCSAT bit is then used to select the type of saturation - either Normal Saturation or Super Saturation. The same saturation mode will apply to both accumulators, if both SATA and SATB are 1.

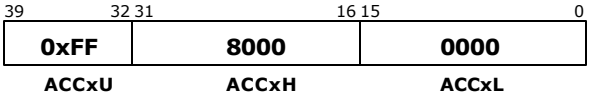
Lastly, there is a SATDW bit in the CORCON which enables Data Write Saturation. This feature allows for the “best possible” 16-bit value to be stored from the accumulator when the SAC instruction is used. We will shortly see how this feature is used.

Normal Saturation (1.31)

- ACCxU is always a sign extension of ACCx<31>
- Correct resultant sign bit is always preserved during operations
  - Provides “Signal Clipping” not Sign-Wrapping
- Largest positive value is **0x007FFFFFFF** (~+1.0)



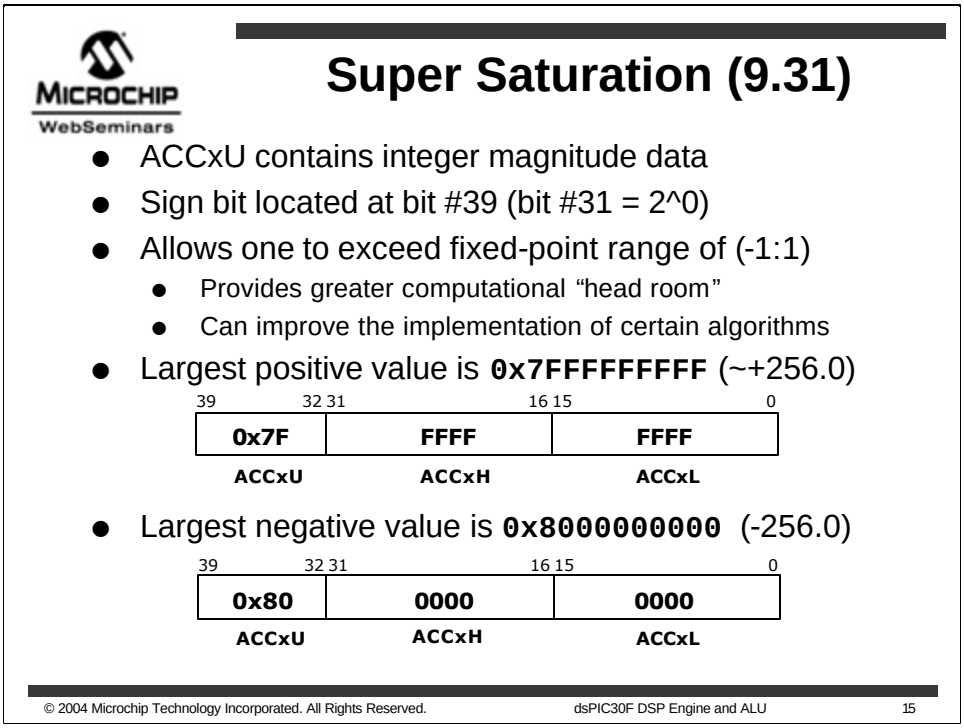
- Largest Negative Value is **0xFF80000000** (-1.0)



The Normal Saturation mode supports a 1.31 *(pronounced as “one dot thirty-one”)* data format. The 1.31 data format uses just one magnitude bit (which is a sign bit) and 31 fractional bits to represent a value.

When Normal Saturation is enabled, the output of the Adder is limited to a range of 32 bits, and Accumulator Upper contains the sign-extension of the 32-bit value stored in Accumulator High and Accumulator Low. The sign of the 32-bit value, stored in Accumulator High bit 15, is extended to all bits in Accumulator Upper. This means that if an operation destroys the sign-bit stored in Accumulator bit 31, the Adder will saturate the result to either the largest 32-bit negative number or the largest 32-bit positive number.

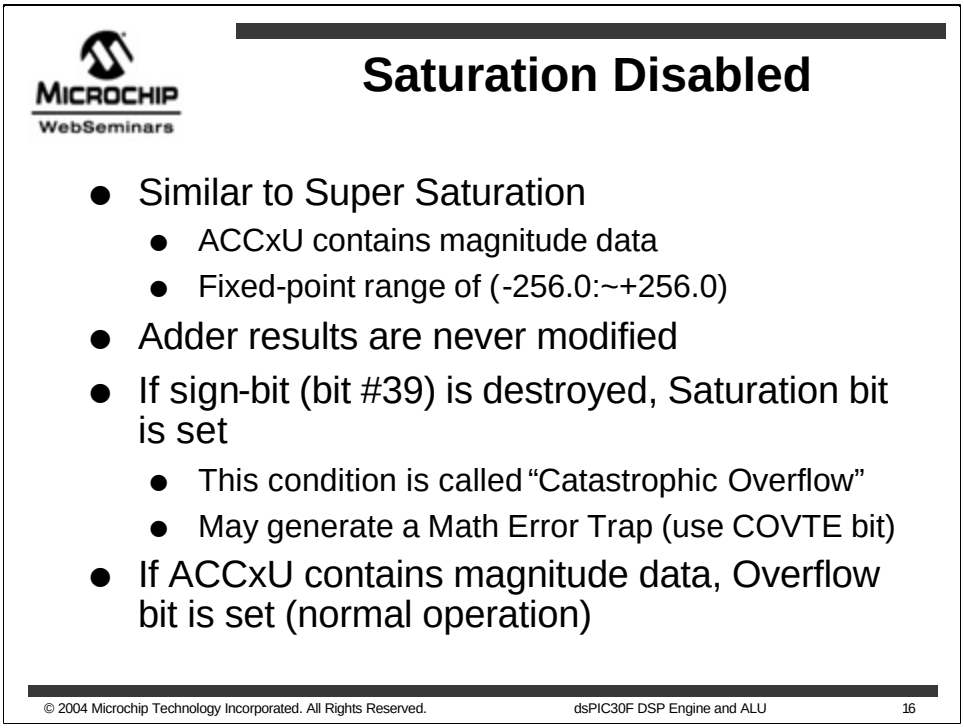
In this example, even though this operation resulted in saturation, it did not result in an overflow, and the resulting OB status bit is zero. Since Normal Saturation restricts results to 32-bits, the upper guard bits in Accumulator Upper will always contain only the sign information, either all 0’s or all 1’s. As a result, overflow can never occur in Normal Saturation mode.



When Super Saturation is enabled, the output of the Adder is limited to a range of 40 bits, and the Accumulator Upper register contains the upper 8-bits of magnitude information. This data format is referred to as 9.31 *(pronounced as "nine dot thirty-one")*, because it has 9 magnitude bits, which includes one sign bit, and 31 fractional bits.

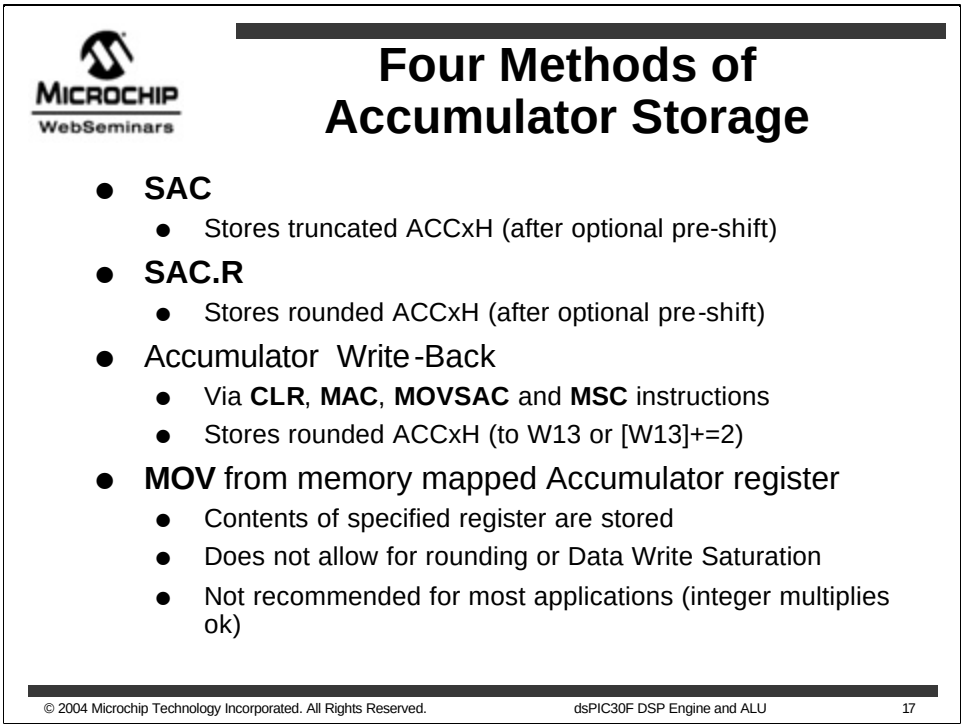
In Super Saturation mode, when an add operation destroys the sign-bit stored at Accumulator bit 39, the Adder will saturate the result to either the largest 40-bit negative number or the largest 40-bit positive number. The maximum negative Accumulator value is 0x8000000000 *(pronounced as "hex 8 billion")* or -256.0 *(pronounced as "minus two-hundred fifty-six point zero")* as a fixed-point value. The largest positive Accumulator value is 0x7FFFFFFFFF *(pronounced as "hex 8 billion minus 1")* or approximately +256.0 *(pronounced as "positive two-hundred fifty-six point zero")*.

The Super Saturation mode provides computational head room which may be very useful in algorithms, while simultaneously providing system protection from runaway calculations. The accumulator value can increase and decrease as long as the value is within the range of -256 *(pronounced as "minus two-hundred fifty-six")* and +256 *(pronounced as "positive two-hundred fifty-six")*.



When saturation is disabled, the output of the Adder is never modified. In this mode, the Adder provides the same dynamic range as with Super Saturation mode, namely -256.0 to approximately +256.0, but no clipping will occur when this range is exceeded.

When an operation exceeds the allowed range, the lower 40-bits of the result will remain unchanged, even though they would be incorrect. This condition is called a “Catastrophic Overflow”, and the user has the option of having this generate a Math Error Trap. This selection is controlled by the Catastrophic Overflow Trap Enable bit, which resides in the INTCON1 register.

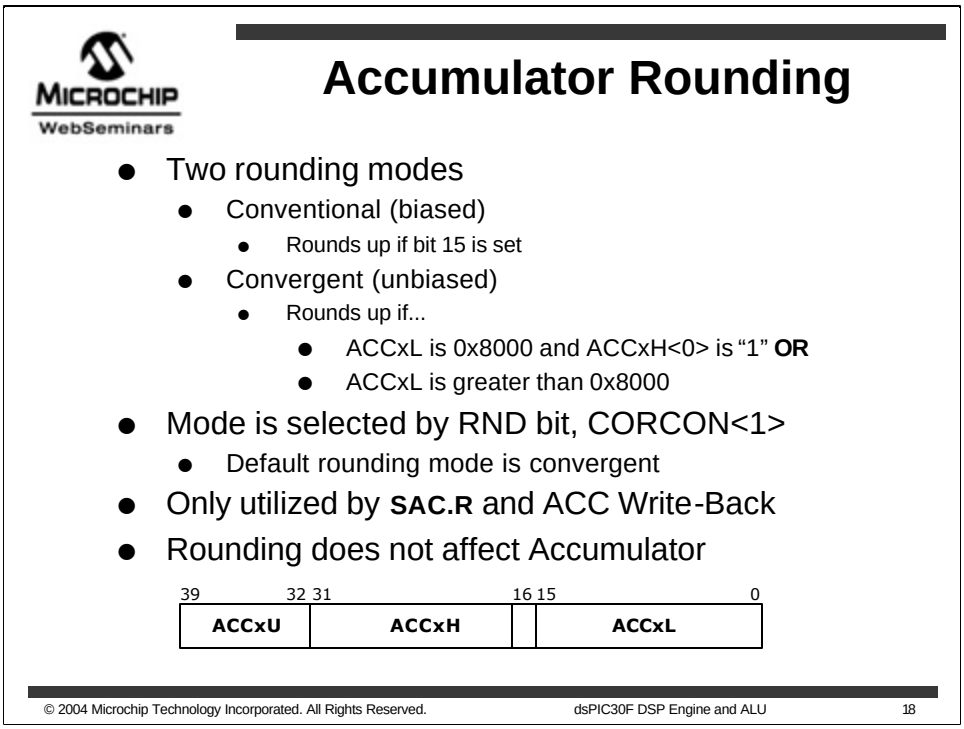


Thus far we've talked about the 17-by-17-bit Multiplier and Adder, which load data into the 40-bit accumulators, and now we are going to see how data is stored out of the accumulators. The dsPIC30F provides four methods for saving data from the accumulators.

The SAC and SAC.R *(pronounced as "sack and sack dot R")* instructions provide the most convenient and most straightforward methods of saving the accumulator. These instructions store the contents of an accumulator to a location in memory, using either direct or indirect addressing. Accumulator High is the accumulator target source register, and one may optionally shift the data as it is being stored. We'll see an example of these instructions in the upcoming foils.

The Accumulator Write-Back is a special feature which may be used only with several DSP instructions. This method of storage allows one to use W13 or "W13-indirect-with-post-increment" as the target destination of the accumulator move, but does not allow for the data to be shifted as it is stored.

Lastly, since the six accumulator registers are memory mapped, one can also store the contents of an accumulator register using the MOV instruction.



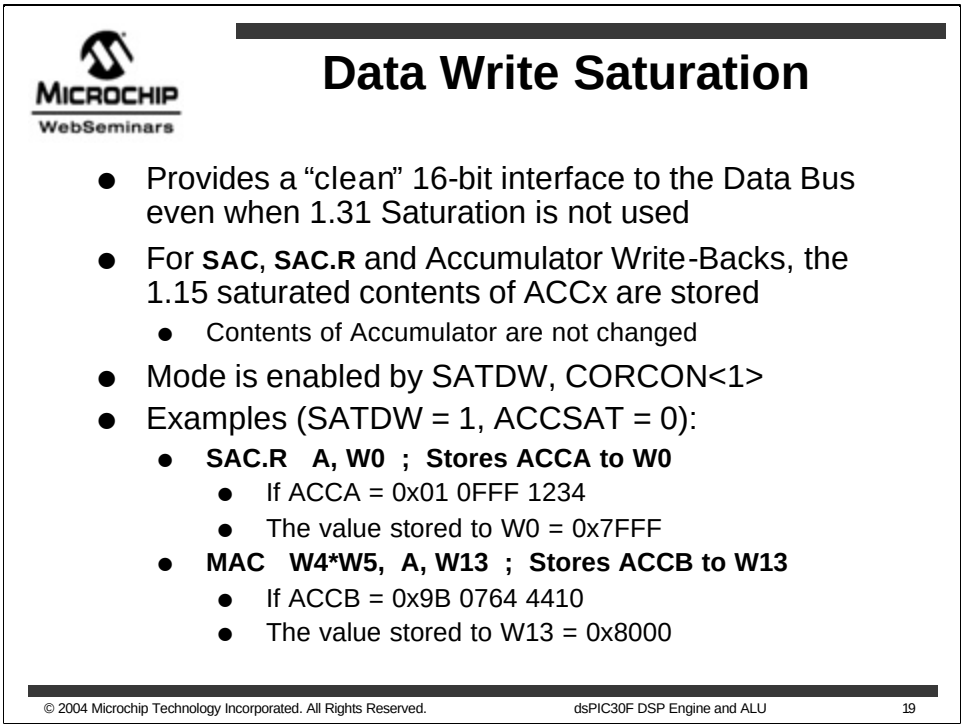
The DSP Engine provides two data rounding modes: Conventional (or biased) rounding and Convergent (or unbiased) rounding. The rounding logic is only utilized during execution of the SAC.R instruction and also when an Accumulator Write-Back is performed.

Conventional rounding is the typical rounding that most people think of, and values are rounded up whenever bit 15 of Accumulator Low is set. This rounding method is considered "biased" rounding, because over time, the effect of always rounding up when Accumulator Low equals 0x8000 *(pronounced as "hex 8 thousand")* slightly skews results to the high side.

A more statistically accurate method of rounding is Convergent rounding, which removes the bias of Conventional rounding. Unbiased rounding rounds up only if Accumulator Low is greater than 0x8000, or if Accumulator Low equals 0x8000 and bit 16 in the Accumulator High register is 1.

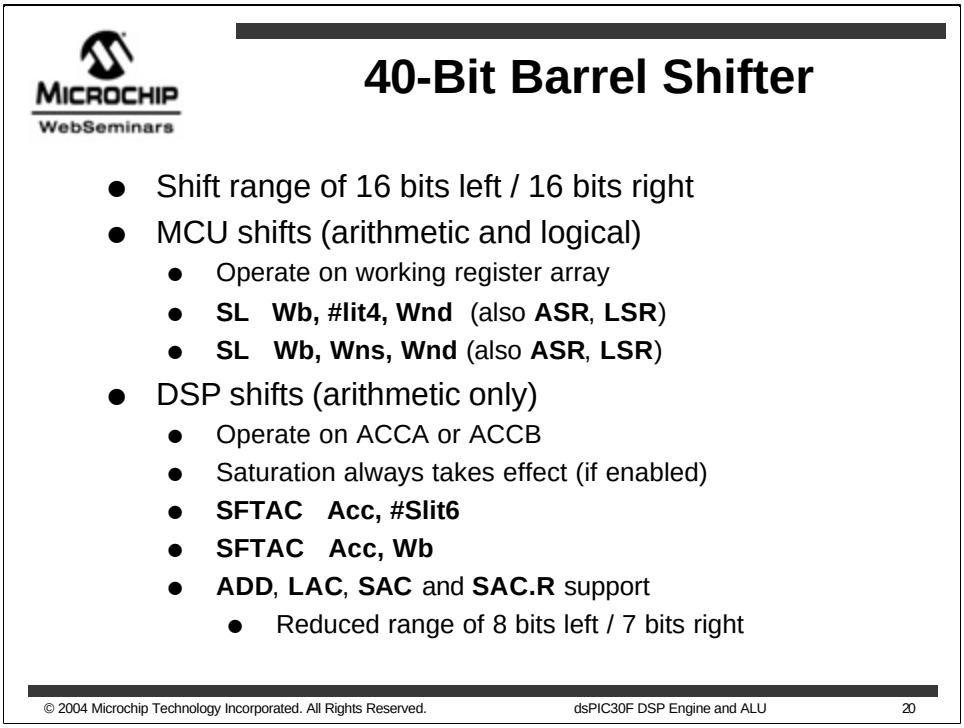
As you can see, the only difference between the two modes is when Accumulator Low is 0x8000. With Conventional rounding, the value will always be rounded up, but in Convergent rounding, the value will only be rounded up statistically half of the time. In most applications, the method of rounding will have little impact, yet both methods are provided. Even though Convergent rounding is more accurate, Conventional rounding is provided to support "bit accurate" algorithms which utilize traditional rounding.

A final point to understand about rounding is that rounding only occurs while data is being stored back through the X Data Bus. This implies that when a value is rounded, it does not modify the value stored in the accumulator.



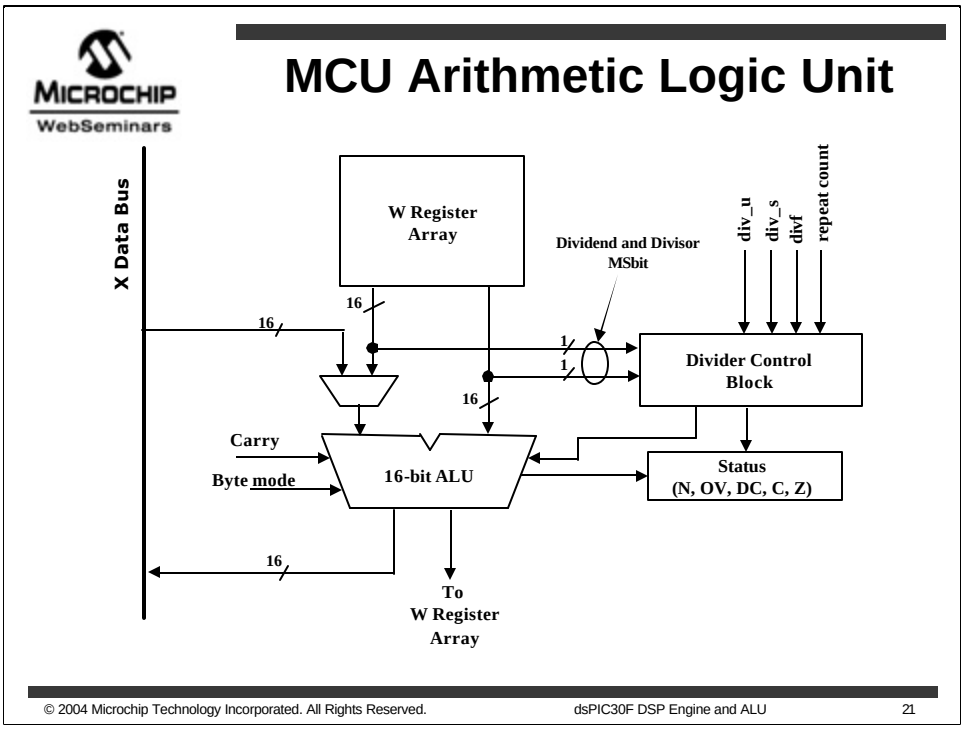
Data Write Saturation is a useful feature when Normal Saturation is not in use. As we have seen earlier, when Super Saturation is enabled or Saturation is disabled, the guard bits in Accumulator Upper are utilized to store magnitude data. Yet, when one stores the contents of the accumulator with SAC, SAC.R or with Accumulator Write Back, the source is the Accumulator High register. If no shifting is performed, Accumulator Upper is ignored when the value is stored to memory. To accommodate for this situation, Data Write Saturation will store the best possible 16-bit value from the 40-bit accumulator.

In the first example, the value stored to W0 is 0x7FFF, since Accumulator A is positive and larger than 32-bits. In the second example, the value stored to W13 in the Accumulator Write Back of the MAC instruction is 0x8000, since Accumulator B is negative and larger than 32-bits.



The 40-bit Barrel Shifter is used to shift data, with a shift range of up to 16-bits to the left or 16-bits to the right. The Barrel Shifter is shared between the MCU and DSP instructions.
When an MCU shift is performed, the source and destination must come from the working register array. When a DSP shift is performed, the source must be an Accumulator, and the destination must either be the same accumulator, any W register or any memory location.

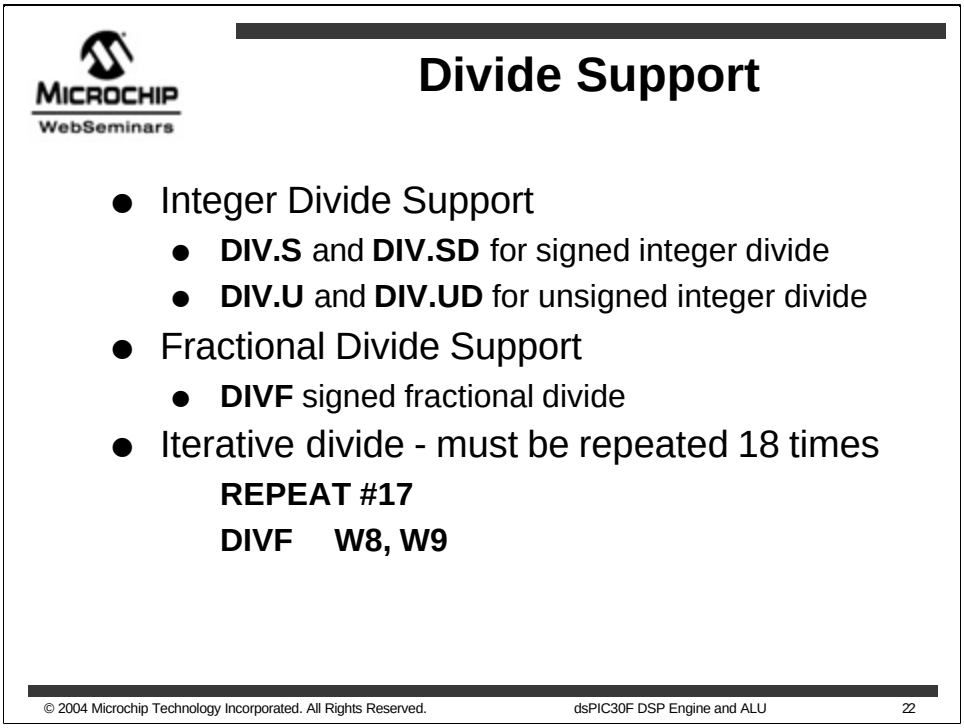
For MCU shifts, the “N” and “Z” status bits are updated so that the user can determine if the result is negative or zero. For DSP shifts, the resultant value is subject to saturation before it is stored. Additionally, if the shift range of the SFTAC exceeds plus or minus 16, a Math Error Trap will be generated.



A top-level view of the 16-bit Arithmetic and Logic Unit, or ALU, is illustrated here. We can see that the ALU interfaces with both the X Data Bus and working register array. Either the inputs can come one from Data Memory and the other from a W register, or both inputs can come from W registers. Outputs of the ALU may be stored either to a Data Memory address or to a W register.

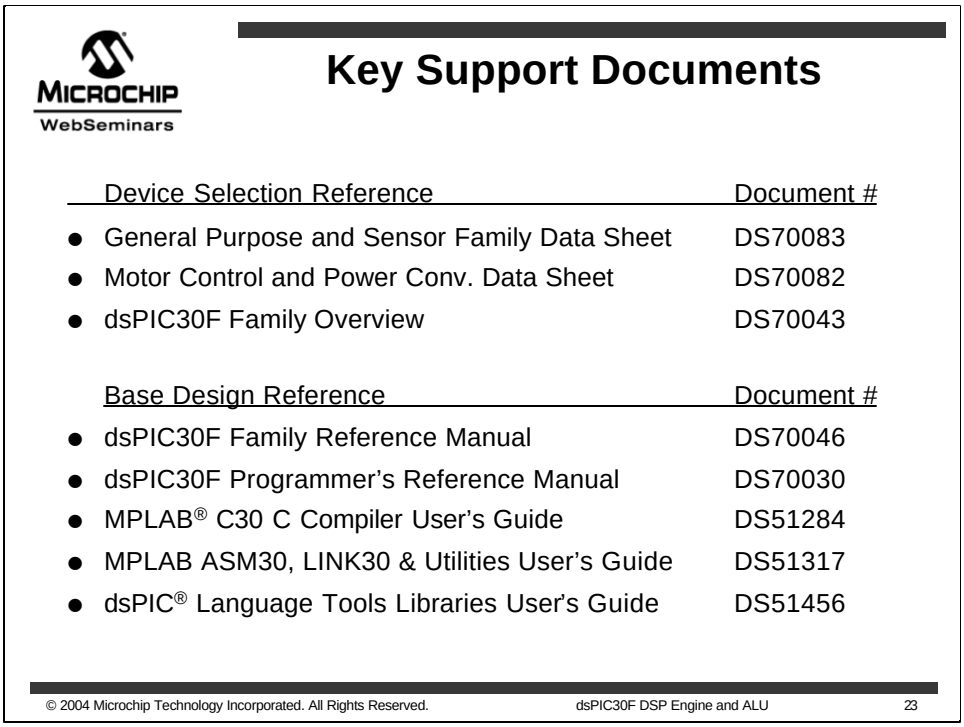
Unlike the DSP Engine, the ALU supports byte wide operations. This feature provides backward compatibility with legacy Microchip PICmicro® devices, and also flexibility for working with byte-sized data. When working in byte mode with the working register array, the lower byte of the specified working registers are utilized, always leaving the upper byte unaffected. Also, the lower byte of the Status Register, consisting of the Negative, Overflow, Digit Carry, Carry and Zero flags, will be updated based on the results of the byte operation.

For details on all the arithmetic and logical instructions supported by the dsPIC30F architecture, please refer to the dsPIC30F Programmer's Reference Manual.



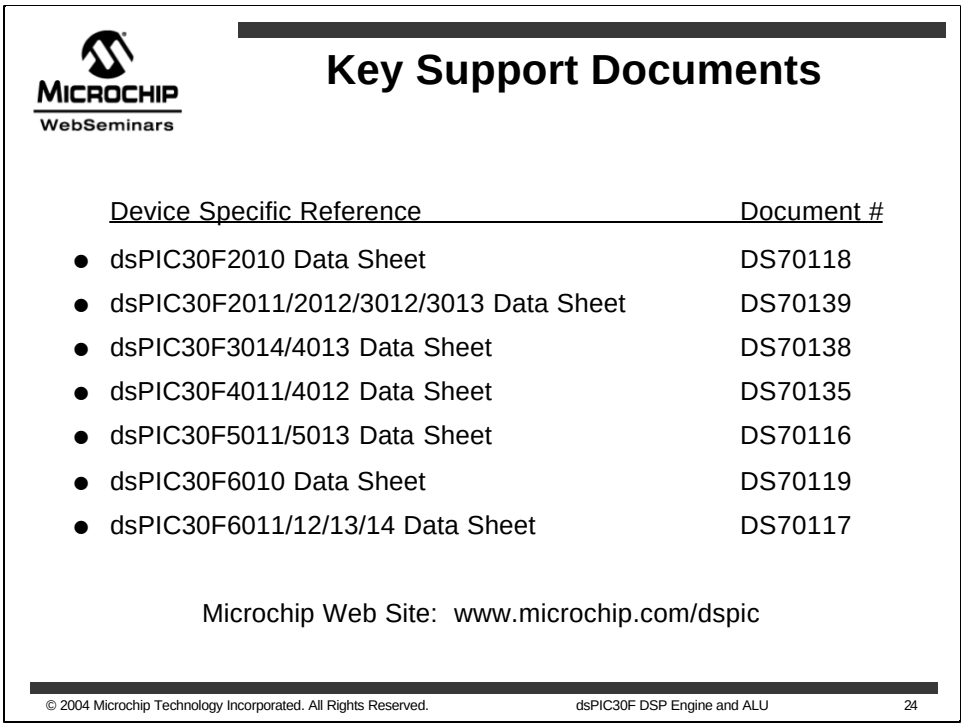
The Divide Control Block provides support for signed-integer or unsigned-integer division, through the DIV.S and DIV.U instructions *(pronounced as "DIV dot S and DIV dot U")* respectively, as well as fractional division through the DIVF instruction. The integer divide instructions also support a long divide, where the numerator is 32-bits wide and stored in two adjacent working registers, but the denominator is 16-bits wide. The long divide instructions use the DIV.SD and DIV.UD *(pronounced as "DIV dot SD and DIV dot UD")* notation.

Unlike all other dsPIC30F instructions, the divide instructions must be used in conjunction with the REPEAT instruction. Specifically, the divide instruction must be executed 18 times within a REPEAT loop, as shown in the example. In this example, W8 is divided by W9. The quotient will be stored in W0, and the remainder in W1.



For more information, here are references to some important documents that contain a wealth of information about the dsPIC30F family of devices.

The Family Reference Manual contains detailed information about the architecture and peripherals, whereas the Programmer's Reference Manual contains a thorough description of the instruction set.



For device-specific information such as pinout diagrams, packaging and electrical characteristics, the device datasheets listed here are the best source of information.

All these documents can be obtained from the Microchip web site shown, by clicking on the "dsPIC® Digital Signal Controllers" or "Technical Documentation" link.

Well… this wraps up the seminar on the dsPIC30F DSP Engine and ALU. Thank you for your interest in the dsPIC30F Family of Digital Signal Controllers.