An Introduction to Programming and Visual Basic for Applications

Data and Information

- Data are raw facts
- Examples of data include transactions, dates, amounts, etc.
- Information are data that have been processed into a usable form
- Information includes tables, documents, charts, etc.
- Goal of computer applications is to process data into information

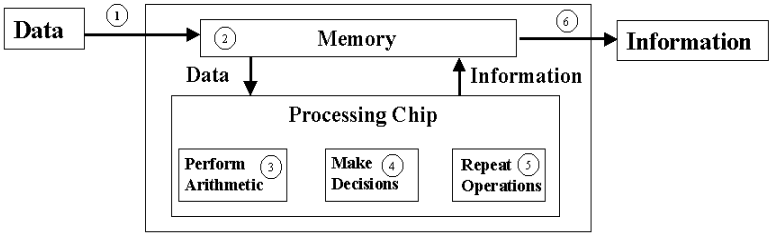
Six Basic Computer Operations

1. A computer can receive (input) data
2. A computer can store data in memory
3. A computer can perform arithmetic and manipulate text strings
4. A computer can compare the contents of two memory locations and select one of two alternatives
5. A computer can repeat a group of operations
6. A computer can output information (processed data)

We will use pseudocode statements to demonstrate these six operations.

3

Computer Operations



4

Programs and Programming

A program is a very specific set of rules that tell the computer which switches should be "ON" or "OFF".

The process of creating a program is called programming.

The computer only knows what it is told through programs, so they must be accurate and very specific.

5

What is Programming?

- Deciding if there is a task to be accomplished or problem to be solved using a computer, e.g., is there a need for a program?
- Determining the nature of the task or problem, e.g., what must the program do?
- Developing a plan that will accomplish the task or solve the problem, e.g., generating the step-by-step process that the program will follow (algorithm).
- Converting the plan into a computer language program
- Testing the program to ensure it accomplishes task or solves problem defined earlier.
- Implementing the program to accomplish the task or solve the problem.

6

THE "IDEA" OF A PROGRAM

Computer programs are like road directions.
Suppose I want to tell someone how to drive from point *A* to point *B*. Here's an example:

1. Leave parking lot and turn right onto Winter Street.
2. Drive 1 mile.
3. Turn right onto Route 30 East.
4. Drive 0.5 mile.
5. Turn left onto Wellesley Street.
6. Drive 0.8 mile.
7. Turn left onto Chestnut Street.

A person could follow these directions in a step-by-step fashion and arrive at Chestnut Street with no problem.

Suppose that we switched two of the instructions:

1. Leave parking lot and turn right onto Winter Street.
2. Drive 1 mile.
3. **Turn left onto Wellesley Street**.
4. Drive 0.5 mile.
5. **Turn right onto Route 30 East**.
6. Drive 0.8 mile.
7. Turn left onto Chestnut Street.

- As a consequence, the individual would become hopelessly lost.
- To be effective, road directions must be unambiguous and totally logical

- The same holds true for computer programs.
- A *program* is a sequence of unambiguous instructions that the computer implements one after another in a mindless manner.
- These instructions are sometimes called *statements*.
- For example, a very simple VBA Sub program to add two numbers can be written as

```
Sub Adder()
    a = 10
    b = 33
    c = a + b
End Sub
```

It's not too difficult to see that this program assigns two numeric constants to the variables **a** and **b**, and then adds them and assigns the result to the variable **c**.

Adder Good

Suppose you redid the program and switched two of the statements:

```
Sub Adder()
    a = 10
    c = a + b
    b = 33
End Sub
```

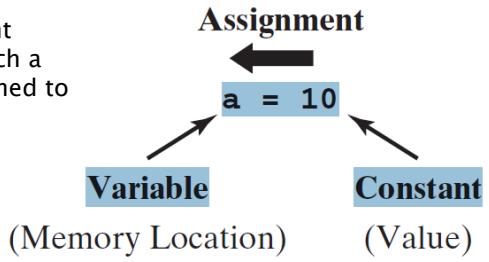
- The program performs the addition before assigning a value to **b**.
- Therefore, **c** would have a value of 10, rather than the desired result: 43.
- The computer is like an *idiot savant* who can implement instructions at an incredibly fast rate.
- But if your instructions to it (i.e., the program) are ambiguous and illogical, it won't work properly

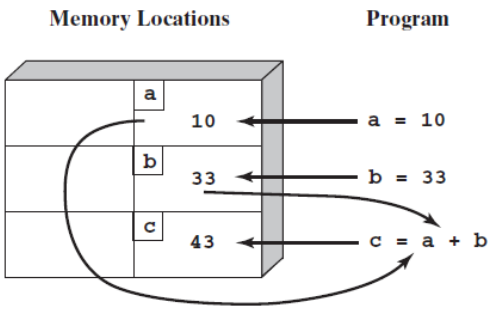
Adder Bad

THE CONCEPT OF ASSIGNMENT

- I have used terms like "constants," "variables," and "assignment."
- What do these mean?
- There are two fundamental ways in which information is represented in a computer language like VBA: directly as constants and symbolically as variables.
- A *constant* is a value that does not change.
- In contrast, a *variable* is a symbolic name that can take on different values.
- They are related by the fact that constants can be assigned to variables

A VBA assignment statement in which a constant is assigned to a variable



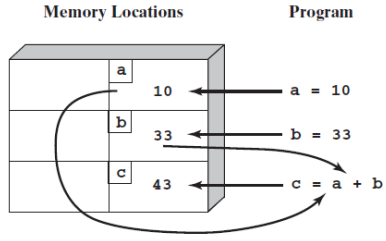


Assignment of constants and variables to locations in the computer's memory.

- The variable is the label that the computer associates with the memory location where it stores a constant value.
- As depicted in the figure, the first two statements,

  a = 10
  b = 33
- cause the computer to set up two storage locations in its memory, which it labels **a** and **b**, and into which it stores the constants 10 and 33.
- The third statement,

  c = a + b
- instructs the computer to add the stored values of the variables **a** and **b**, and to assign the result to the variable **c**.
- The equals sign in the assignment statement can be thought of as meaning "is replaced by" or "is assigned the result."
- ***It is critical to recognize that this use is different from the conventional one wherein the equals sign signifies equality.***



Assignment Statements

- Assignment statements must always have a single variable on the left side of the equals sign.
- Because they both violate this rule, the following legitimate algebraic equations would be invalid as VBA statements:

  a + b = c
  3 = x – y + 99
- However, although the following statement is not a proper algebraic equation, it is a perfectly valid assignment statement:

  x = x + 1
- After this statement was executed, the memory location for the variable **x** would have a value that was one higher than its previous value.

DECISIONS AND LOOPS

- We understand that a program is a *sequence* of unambiguous instructions that the computer implements one after another.
- In reality, this should be amended to read "A program is a sequence of unambiguous instructions that the computer implements one after another, ***unless you instruct it to do otherwise***."
- There are two ways in which you can make a program deviate from its sequential mode of operation:
  - **decisions and loops**.
- This allows you to vastly increase a program's power to solve numerical problems.
- Computer scientists have proven that any numerical calculation can be implemented by some combination of sequences, decisions, and loops.

- A *decision* involves implementing a statement, depending on the outcome of a decision.
- The *If/Then* statement is the simplest means of implementing decisions in VBA.
- Suppose that you want to determine the absolute value of a number.
- If the number is greater than or equal to zero, you do not have to do anything.
- However, for a negative number, you would have to make the sign positive.
- An If/Then statement can be used for this purpose, as in the code

```
If a < 0 Then
   a = -a
End if
```

- VBA is designed to be easy to understand, the meaning of this If/Then statement is pretty straightforward.
- You can read it as if it were an English sentence: "If **a** is less than zero, then set **a** equal to its negative."
- Because of the nature of assignment, this results in the absolute value being generated and stored in the memory location for **a**.

- In addition to generating a single action, a decision can be used to implement one action if a statement is true and another if it is false.
- Suppose that you want to determine the sign of a number.
- The following *If/Then/Else* statement can be used for this purpose:

```
If a < 0 Then
   s = -1
Else
   s = 1
End if
```

  - If **a** is less than zero, **s** will be assigned a value of -1 which is mathematically equivalent to a negative sign.
  - If **a** is not greater than zero (that is it is less than or equal to zero), **s** will be assigned a value of 1.

Loops

- A *loop* repeats a VBA statement (or statements) several times.
- The *For/Next loop* is the simplest way to do this.
- Suppose that you want to calculate the sum of the first **n** positive whole numbers.
- A For/Next loop can be used for this purpose:

```
x = 0
For i = 1 to n
    x = x + i
Next i
```

- Because VBA is so easy to understand, you should be able to deduce what's going on here.
- Suppose that **n** is assigned a value of 5.
- After **x** is assigned a value of 0, the actual loop begins with the For statement, wherein the variable **i** is initially set to 1.
- The program then moves to the following line, in which **x + i** is assigned to **x**.
- Since **i** is equal to 1, **x** would then be equal to 1.
- At the Next statement, the computer increments **i** by 1 so that **i** becomes 2.
- The computer then transfers control back to the For statement, where VBA determines whether **i** is greater than **n** (i.e., 5).
- If so, VBA exits the loop by transferring to the line immediately following the Next statement.
- If not, it repeats the body of the loop, so that **x** becomes 3.
- The process is repeated until **i** is greater than **n**.
- In our example, **x** would be equal to 15 and the loop would be terminated.

Looping

Types of Computer Languages

Procedural: Monolithic programs that run from start to finish with no intervention from user other than input

- Basic, QBasic, QuickBasic
- COBOL
- FORTRAN
- C

Object Oriented/Event Driven (OOED): Programs that use objects which respond to events; use small segments to code for each object

- JAVA
- Visual Basic
- **Visual Basic for Applications (VBA)**
- Visual C++

19

Levels of Computer Languages

Low Level: at the level of the computer, i.e., in binary (0-1) format Computer can only execute a binary form of a program

Intermediate Level: close to the computer but uses English words or mnemonics, e.g., Assembler, that is converted directly into binary

High Level: at the level of the programmer using English words and clearly defined syntax; must be converted or translated into binary for computer to implement it, e.g., VBA

Need a software program to handle the conversion of high-level into binary

20

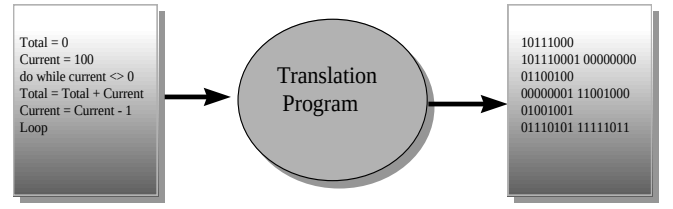
Translating from High-level Language to Binary

Interpreted: each statement translated as it is executed--slow but easy to use

Compiled: entire program is converted to binary--executes faster, but more difficult to use (.exe files are compiled programs)

VBA is interpreted during creation and testing but can then be compiled into an .exe file

21



High level language program

Machine language progam

22

Why Use VBA in Excel 2010?

- Microsoft Excel 2010 is an powerful tool that you can use to manipulate, analyze, and present data.
- Despite the rich set of features in Excel, you might want to find an easier way to perform a mundane, repetitive task, or to perform some task that the UI does not seem to address.
- Office applications like Excel have Visual Basic for Applications (VBA), a programming language that gives you the ability to extend those applications

- VBA works by running *macros*, step-by-step procedures written in Visual Basic
- Learning even a small amount of VBA code makes your work easier and gives you the ability to do things in Office that you did not think were possible.
- Once you have learned some VBA, it becomes much easier to learn a whole lot more—so the possibilities here are limitless.

- The most common reason to use VBA in Excel is to automate repetitive tasks.
- For example, suppose that you have a few dozen workbooks, each of which has a few dozen worksheets, and each of those needs to have some changes made to it.
- The changes could be as simple as applying new formatting to some fixed range of cells
- or as complex as looking at some statistical characteristics of the data on each sheet, choosing the best type of chart to display data with those characteristics, and then creating and formatting the chart accordingly.

- VBA is not just for repetitive tasks though.
- You can also use VBA to build new capabilities into Excel
- You could develop **new algorithms** to analyze your data, then use the charting capabilities in Excel to display the results and to perform tasks that integrate Excel with other Office applications such as Microsoft Access 2010.
- Of all the Office applications, Excel is the one most used as something that resembles a general development platform.
- In addition to all the obvious tasks that involve lists and accounting, developers use Excel in a range of tasks from **data visualization** to software prototyping.

Using Code to Make Applications Do Things

- Writing code is not mysterious or difficult, the basic principles use every-day reasoning and are quite accessible.
- The Office 2010 applications are created in such a way that they expose things called *objects* that can receive instructions.
- You interact with applications by sending instructions to various objects in the application.
- Objects are varied, and flexible, but they have their limits.
- They can only do what they are designed to do, and they will only do what you instruct them to do.

Object-Oriented Event-driven Programming (OOED)

OOED uses **objects**, or self contained modules that combine data and program code which pass strictly defined messages to one another.

OOED is easier to work with, because it is more intuitive than traditional programming methods.

Visual Basic is an **OOED** language.

Users can combine the objects with relative ease to create new systems or extend existing ones.

Properties of objects are **attributes** associated with an object.

Methods of objects are those activities that the object can carry out.

Objects respond to **events.**

28

Objects

- Programming objects relate to each other systematically in a hierarchy called the *object model* of the application.
- The object model roughly mirrors what you see in the user interface;
  - for example, the Excel object model contains the **Application**, **Workbook**, **Sheet**, and **Chart** objects, among many others.
- The object model is a conceptual map of the application and its capabilities.

Properties and Methods

- You can manipulate objects by setting their *Properties* and calling their *Methods*.
- Setting a property changes some quality of the object.
- Calling a **method** causes the object to perform some action.
  - For example, the **Workbook** object has a **Close** *method* that closes the workbook, and an **ActiveSheet** *property* that represents the sheet that is currently active in the workbook.

Macros and the Visual Basic Editor

- To call object methods and setting object properties you must write your code in a place and in a way that Office can understand;
  - typically, by using the **Visual Basic Editor**.
- Although it is installed by default, many users do not know that it is even available until it is enabled on the ribbon.
- As you have learned how to do this in lab – it's very easy.

The Software Development Lifecycle

The **Software (or Systems) Development Life Cycle (SDLC)** is a systematic methodology for planning, designing, and implementing applications.

A process for writing a computer program:

1. Plan application – you must *Define the problem.
2. Create interface
3. Develop logic and **write code** to handle events
4. Test overall project
   A. Run and test the application to verify that it produces intended results
5. Document project in writing and test in overall environment

*No matter how well a program is written, the objective is not achieved if the program solves the wrong problem.

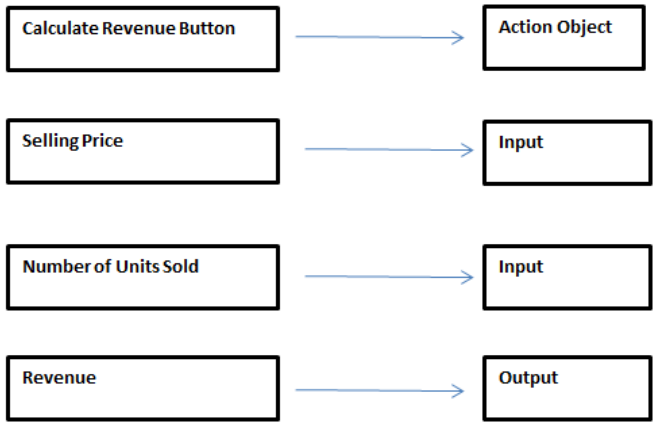
32

Step One: Define Problem

- Before you can create a computer application to solve a problem, you must first clearly define it.
- This may involve a study of the problem to understand the inputs and outputs.
- Must identify the data to be *input* to the program and the results to be *output* from it.
- Sketching an interface is a good way to understand the problem and to communicate your understanding to other people.
- Denote input and output objects as well as *action objects*--those for which code (instructions) are needed.

33

Sketch of Calculate Revenue Interface



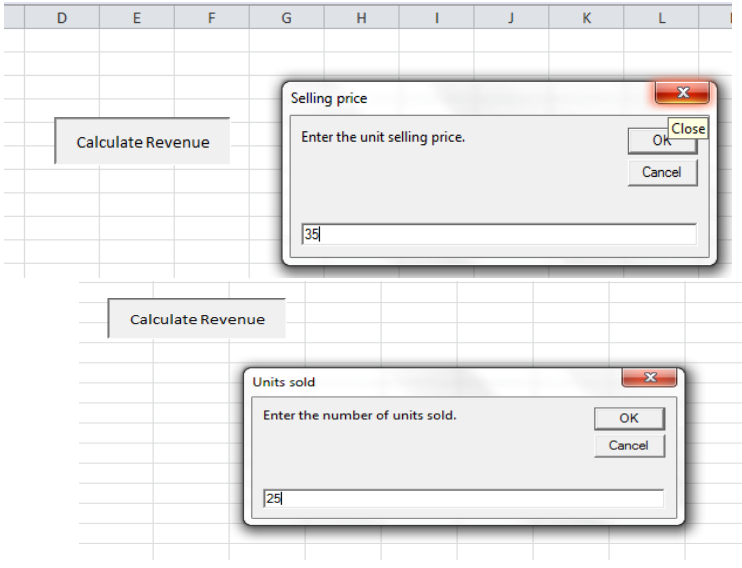
34

Step Two: Create Interface

- Once you have defined problem and sketched interface, you are ready to create interface.
- Doing this with VBA is quite easy.
- For our first program you only need four objects:
  - button for action
  - inputBox for input (Selling Price)
  - inputBox for input (Units Sold)
  - MessageBox for output

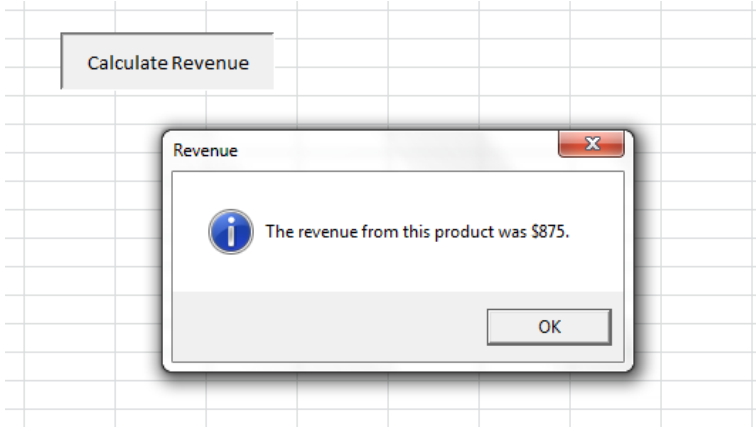
35

Calculate Revenue Interface - Input



36

Calculate Revenue Interface - Output



37

Step Three: Develop Logic for Action Objects

- Need to think about what each action object should do in response to an event
- This is the **logic** for each object
- Use **Input/Processing/Output (IPO) Tables** and **Pseudocode** to develop the logic
- IPO Tables show the inputs, outputs, and the processing to convert inputs into outputs

38

IPO Table for Calculate Revenue Button

Input	Processing	Output
Unit Price Quantity Sold	revenue = unitPrice X quantitySold	Revenue

39

Using Pseudocode

- An important part of the developing the logic for action objects is generating corresponding pseudocode.
- **Pseudocode** involves actually writing a program in English rather than in a computer language.
- When the actual computer program is written, the pseudocode is translated into computer language.
- A pseudocode program is useful for two reasons:
  - The programmer may use it to structure the algorithm's logic in writing.
  - It provides a relatively direct link between the algorithm and the computer program

40

Pseudocode for Calculate Revenue Button

Begin procedure
  Input Selling Price
  Input Quantity Sold
  Revenue = Selling Price x Quantity Sold
  Output Revenue

End procedure

41

Steps Three and Four: Write Code and Test Code

- Once you learn the vocabulary and syntax of a language, you should be able to convert the logic embodied in the pseudocode into a computer language. In our case, we use VBA.
- You need to test the code for each action object as they are created.
- Once the object code is written, the programmer must test and correct it. This stage is referred to as debugging, since it involves removing "bugs".
- Use test data for which you know the correct answer to test the object code.

42

VBA Code for Calculate Revenue Button

```
Sub CalculateRevenue()
    Dim unitPrice As Currency
    Dim quantitySold As Integer
    Dim revenue As Currency

    ' Get the user's inputs, then calculate revenue.
    unitPrice = InputBox("Enter the unit selling price.", "Selling price")
    quantitySold = InputBox("Enter the number of units sold.", "Units
sold")
    revenue = unitPrice * quantitySold

    ' Report the results.
    MsgBox "The revenue from this product was " & Format(revenue,
"$#,##0") _
        & ".", vbInformation, "Revenue"
End Sub
```

43

Step Five: Document and Test Overall Project

- Even with extensive testing, some bugs can often be found in most commercial software.
- With computer software, each program instruction must be absolutely correct. Otherwise, the whole program might fail.
- BE SURE to test your program in the actual environment and on the actual data on which it will be used (just ask IBM at the Olympics).

44

Analyzing and Modifying a VBA Program

Simple Adder Program

```
Sub AdderTwo()
Dim a  As Double
Dim b As Double
Dim c As Double
a = Val(InputBox("Please enter the first value to be added"))
MsgBox "This program is designed to add two numbers"

b = Val(InputBox("Please enter the second value to be added"))

c = a + b
Range("B10").Select
ActiveCell.Value = c
MsgBox "the total  = " & c
End Sub
```

Count High Forty

```
Sub CountCells2()
Dim total As Integer, i As Integer
total = 0

For i = 1 To 4
   Cells(i, 1).Select
   MsgBox "i = " & i

   If Cells(i, 1).Value > 40 Then total = total + 1
Next i

MsgBox total & " values higher than 40"

End Sub
```

```
Sub CountHighSales()
   Dim i As Integer
   Dim j As Integer
   Dim nHigh As Integer
   Dim cutoff As Currency
   cutoff = InputBox("What sales value do you want to check for?")
   For j = 1 To 6
      For i = 1 To 36
         If wsData.Range("Sales").Cells(i, j) >= cutoff Then _nHigh = nHigh + 1
      Next i
      MsgBox "For region " & j & ", sales were above " & Format(cutoff,
"$0,000") _
         & " on " & nHigh & " of the 36 months."
      nHigh = 0
   Next j
End Sub
```

Testing the CountHighSales Program Modifcations

- When testing an application you define particular input values to enter to determine if the application produces the expected output
- How would you test this program?
- What variables could you examine to check the accuracy of the program?

Step Six: Document Project in Writing

- Documentation includes the pictorial and written descriptions of the software. It contains internal descriptions of programming commands and external descriptions and instructions.
- Documentation is necessary since, sooner or later, the program will need to be maintained (correct bugs, add new features, handle problems not thought of previously, etc. This is NOT possible without documentation.
- Written documentation includes books, manuals, and pamphlets that give instructions on software use and discuss the objectives and logic of the software.

50

Risks in Numerical Computing

- Almost all computer calculations involve round off error (limited precision error)
- If not monitored and planned for carefully, such errors can lead to unexpected and catastrophic results
  - Ariane 5 Rocket Failure
  - Patriot Missile Failure during Gulf War
  - Mars Lander Crash

The Explosion of the Ariane 5

- On June 4, 1996 an unmanned Ariane 5 rocket launched by the European Space Agency exploded just forty seconds after its lift-off.
- The rocket was on its first voyage, after a decade of development costing $7 billion. The destroyed rocket and its cargo were valued at $500 million.
- It turned out that the cause of the failure was a software error in the inertial reference system. Specifically a 64 bit floating point number relating to the horizontal velocity of the rocket with respect to the platform was converted to a 16 bit signed integer. The number was larger than 32,767, the largest integer storeable in a 16 bit signed integer, and thus the conversion failed.

- Back

Patriot Missile Failure during Gulf War

- During the Gulf War, an American Patriot Missile battery in Saudi Arabia, failed to track and intercept an incoming Iraqi Scud missile. The Scud struck an American Army barracks, killing 28 soldiers and injuring around 100 other people.
- The General Accounting office reported on the cause of the failure. It turns out that the cause was an inaccurate calculation due to computer arithmetic errors.
- The time in tenths of second as measured by the system's internal clock was multiplied by 1/10 to produce the time in seconds.
- The value 1/10, which has a non-terminating binary expansion, was chopped at 24 bits. The small chopping error, when multiplied by the large number giving the time in tenths of a second, led to a significant error. Indeed, the Patriot battery had been up around 100 hours, and an easy calculation shows that the resulting time error due to the magnified chopping error was about 0.34 seconds. (The number 1/10 equals $1/2^4+1/2^5+1/2^8+1/2^9+1/2^{12}+1/2^{13}+\ldots$.
- A Scud travels at about 1,676 meters per second, and so travels more than half a kilometer in this time. This was far enough that the incoming Scud was outside the "range gate" that the Patriot tracked.

Back

NASAs metric confusion caused Mars orbiter loss

September 30, 1999 CNN

- NASA lost a 125 million Mars orbiter because one engineering team used metric units while another used English units for a key spacecraft operation, according to a review finding released Thursday.
- For that reason, information failed to transfer between the Mars Climate Orbiter spacecraft team at Lockheed Martin in Colorado and the mission navigation team in California.
- Lockheed Martin built the spacecraft.
- People sometimes make errors, said Edward Weiler, NASAs Associate Administrator for Space Science in a written statement.
  - “The problem here was not the error, it was the failure of NASAs systems engineering, and the checks and balances in our processes to detect the error. That's why we lost the spacecraft”