MICROCHIP

Regional Training Centers

Introducing our Project

MICROCHIP
Regional Training Centers

Introducing our Project

- Project Demonstration
- Web Page Development
- HTML Structure and Tags
- Packaging Web Pages

- Lab 2: Web Page Development

Introducing our Project

- Web-Enabled Vending Machine
  - Add network capabilities to existing design
  - Build web page interface
    - Check stock
    - Set products and prices
- Demonstration

MICROCHIP
Regional Training Centers

Web Page Development

- Web Page Development
  - HTML – Page content
  - CSS – Style and layout
- Other Components
  - GIF, JPG, PNG, JavaScript
- Avoid visual editors (large code)

Definition

HTML: Hyper-Text Markup Language
CSS: Cascading Style Sheets

Microchip Regional Training Centers

Sidetrack to HTML

- HTML is a markup language
  - Use tags to indicate sections
  - Tags…
    - are structural
    - wrap content
    - must be /closed
    - can be nested
    - may have attributes with values

```
<tag attr="value">content</tag>
```

MICROCHIP
Regional Training Centers

HTML Structure

- All HTML documents have 3 tags
  - `<html>` wraps entire document
  - `<head>` is header content inside `<html>`
  - `<body>` is viewable content inside `<html>`
- Extra whitespace is ignored
  - Includes new lines
- Case-insensitive
- Syntax is lenient

MICROCHIP
Regional Training Centers

Sample HTML Page

```
<html>
  <head>
    non-displayed header content...
  </head>
  <body>
    displayable content...
    text, images, links...
  </body>
</html>
```

MICROCHIP
Regional Training Centers

HTML Content Tags

- Common <body> tags
  - <p> wraps a paragraph
  - <h1> - <h6> creates headings
  - <a> creates anchors (links)
  - <b>, <i>, <u> provide inline formatting
- Visual elements within <body>
  - <table> creates a table
  - <form> creates a form
  - , <ol> create (un)ordered lists

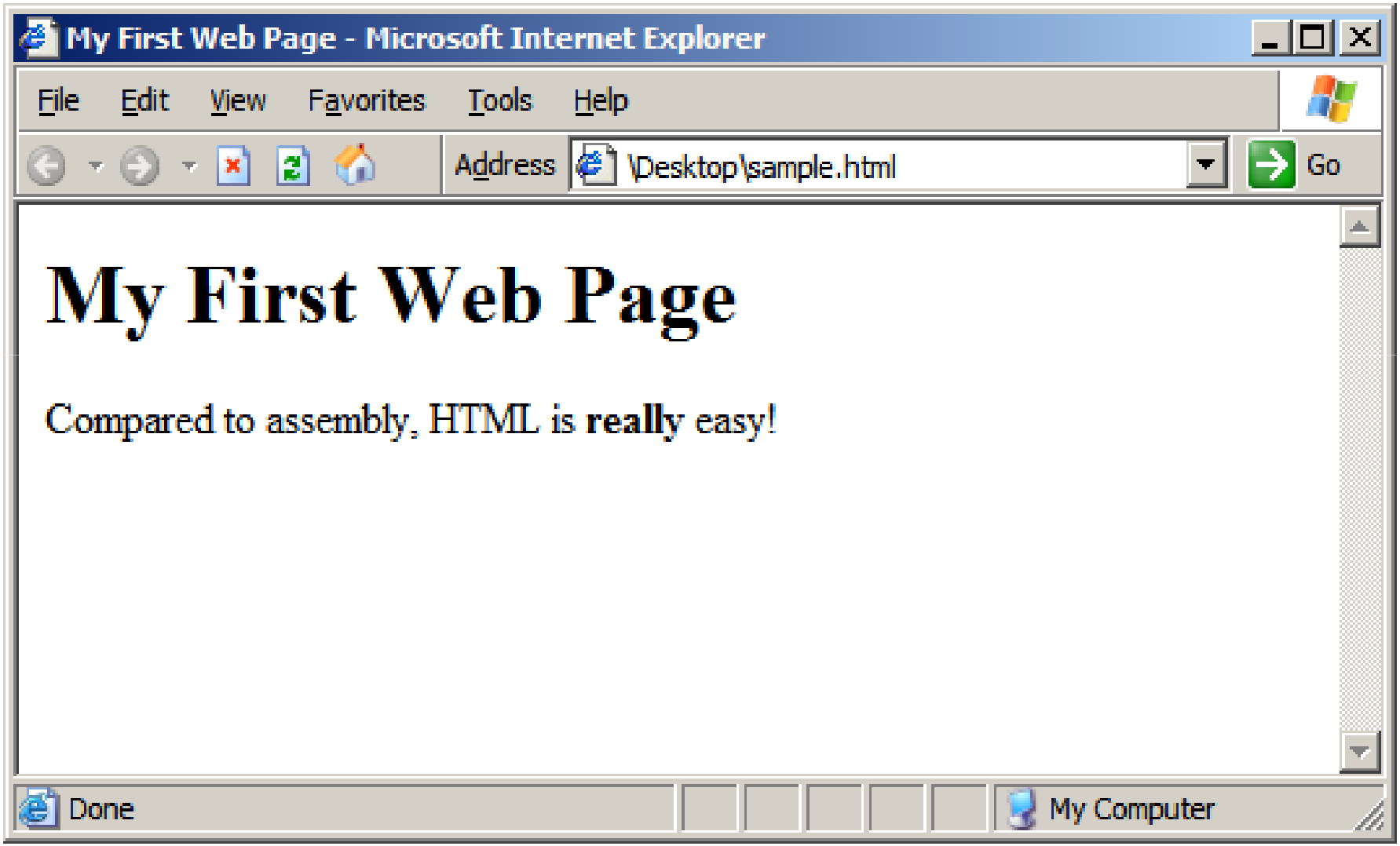
MICROCHIP
Regional Training Centers

Completed Sample Page

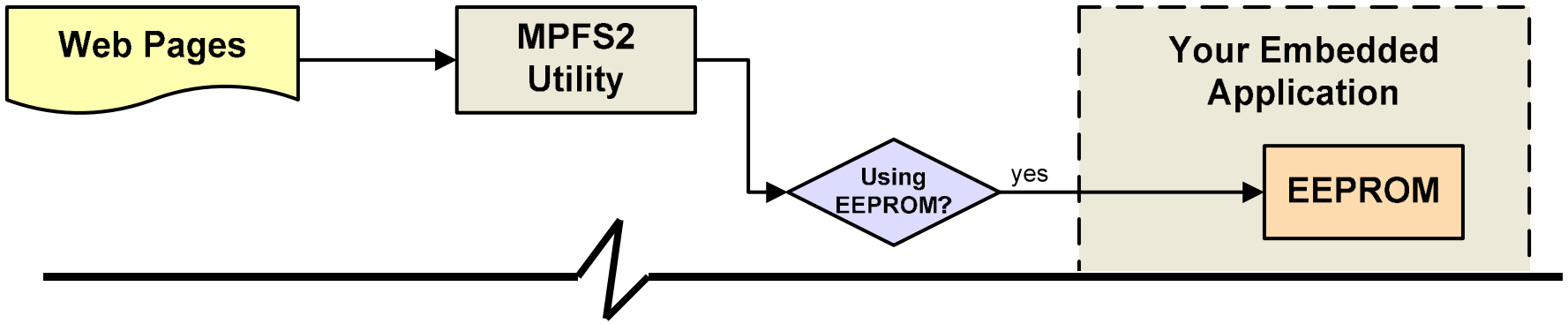
```
<html>
  <head>
    <title>My First Web Page</title>
  </head>
  <body>
    <h1>My First Web Page</h1>
    <p>Compared to assembly, HTML is
      <b>really</b> easy!</p>
  </body>
</html>
```

MICROCHIP
Regional Training Centers

Completed Sample Page



Packaging Web Pages



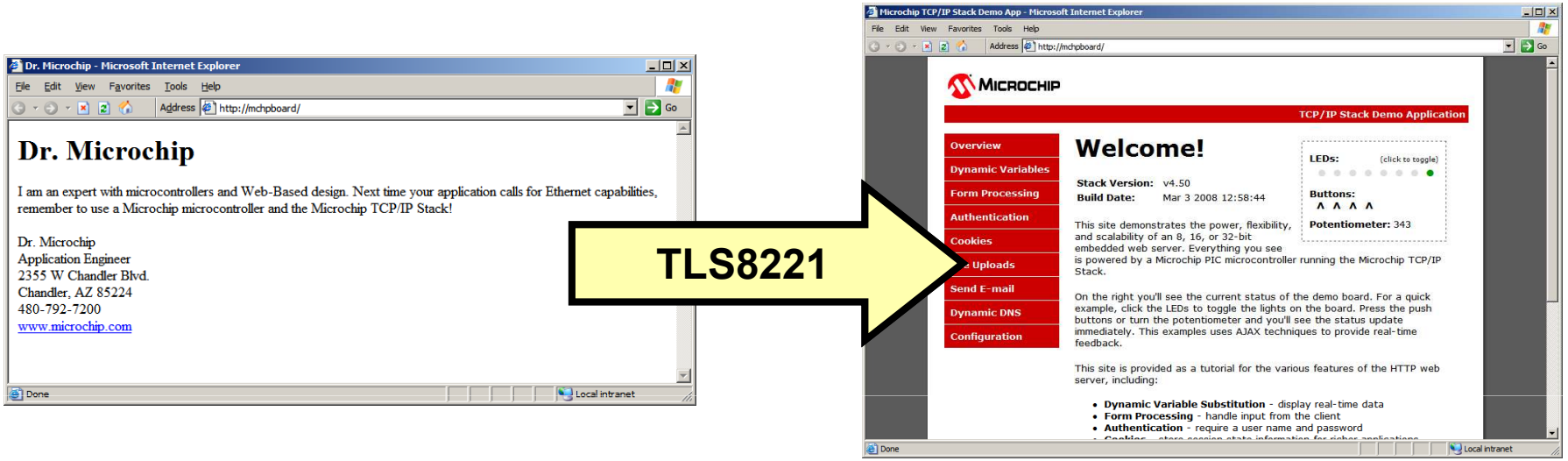
- MPFS2 Utility
  - File system for embedded devices
  - Creates file system image
  - Uploads image to running application
    - Application programs to EEPROM

Lab 2: Web Page Design

- **Goals:**
  - Create a simple web page
  - Test web page locally
  - Compile and upload MPFS2 image
- **Bonus:**
  - Use CSS to format your page

MICROCHIP
Regional Training Centers

Want to Learn More?



- Very basic intro to HTML
- Need to learn more? TLS8221!
  - In depth discussion on HTML
  - Detailed CSS formatting and layout techniques
  - Learn to create web page interfaces

MICROCHIP
Regional Training Centers

Review: The Microchip TCP/IP Stack

- Vending Machine Application
- Principles of HTML
  - Markup language
  - Tag format and structure
- Packaging Web Pages
- Visual Elements with CSS
  - Style and layout

MICROCHIP
Regional Training Centers

Review: Planning for Applications

- Cooperative Multitasking
  - Round-robin task switching
  - Don’t block the processor
- Integrating an Application
  - Copy application into stack
  - Remove blocking code
  - Remove demo app features
- Timing with Tick
  - Timers without blocking loops

MICROCHIP

Regional Training Centers

Implementing Web Applications

MICROCHIP
Regional Training Centers

Building with HTTP2

- Overview of HTTP2
- Dynamic Variables
  - Lab 5: Web-Based Monitoring
- Web-Based Control
  - Lab 6: Control via GET
- Advanced Web-Based Control
  - Lab 7: Advanced Control via POST

Overview of HTTP2

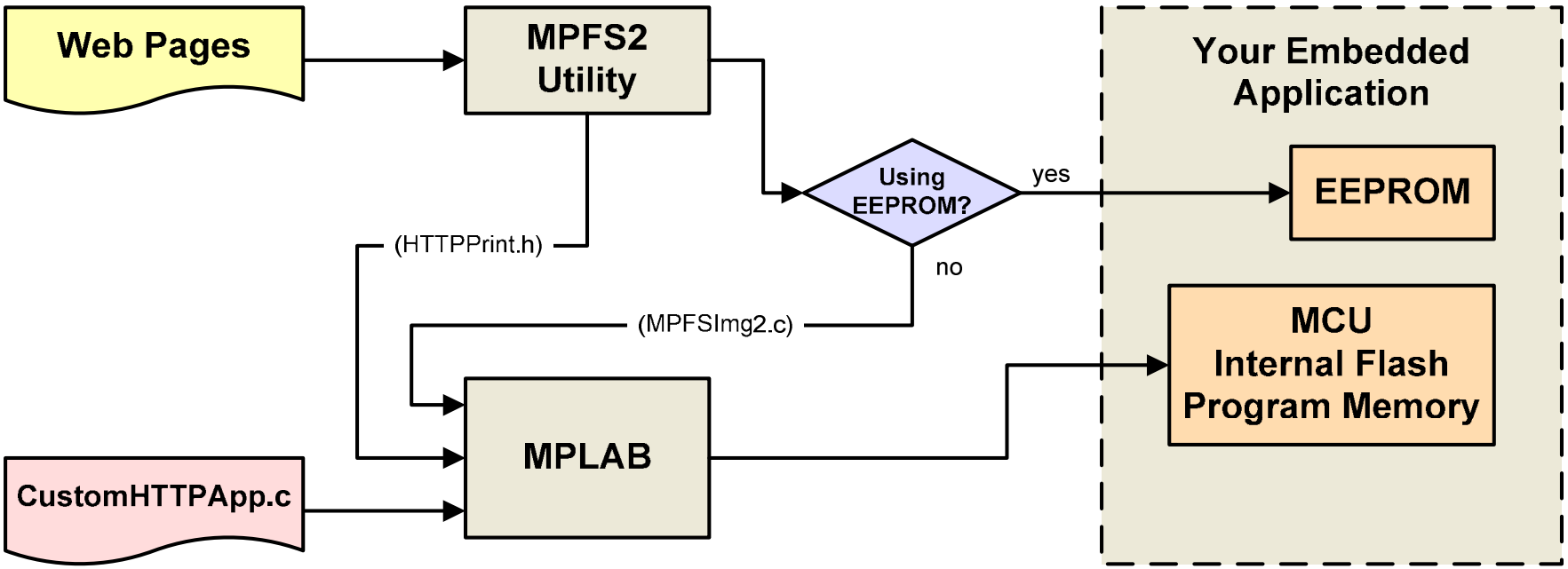
- Standard HTTP Web Server
  - Multiple connections
  - Serves any file type
- Additional Features
  - Dynamic content
  - Form input
  - Authentication
  - Cookies
  - Caching and GZIP Compression

MICROCHIP
Regional Training Centers

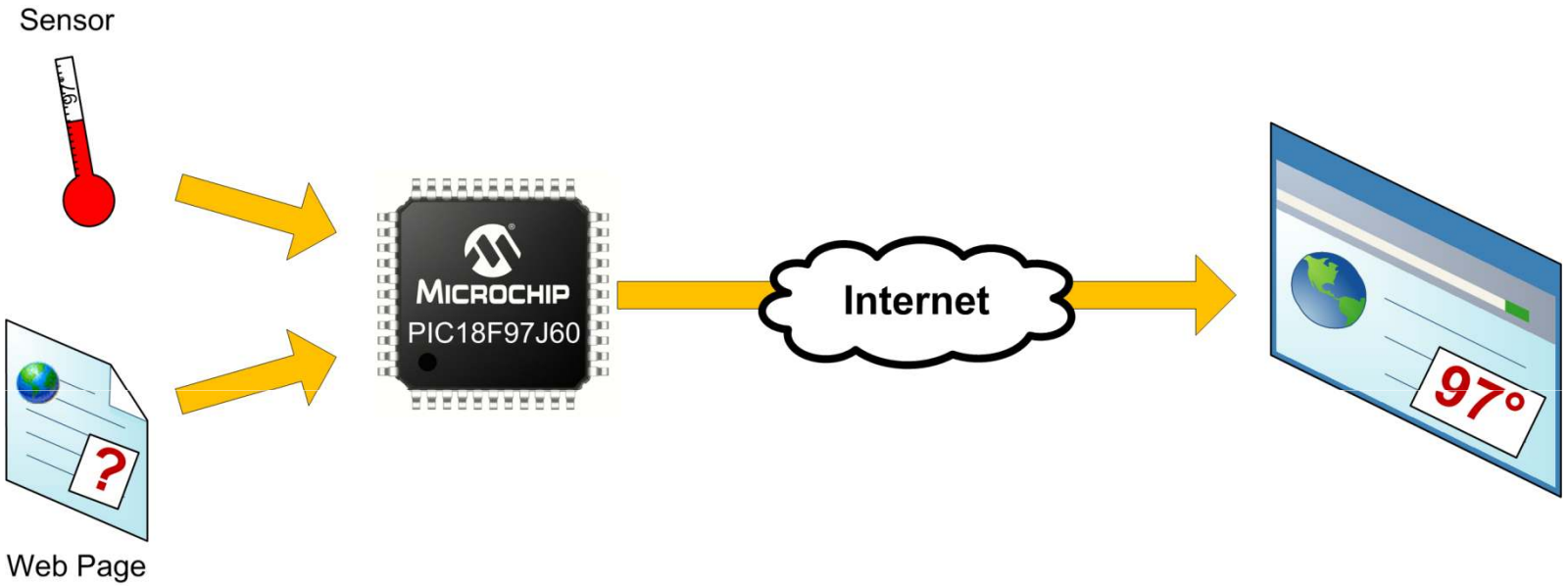
Overview of HTTP2

- MPFS2 File System
  - Internal Flash or external EEPROM / Flash
  - PC Utility builds image file
  - Parses HTML and prepares project
- `CustomHTTPApp.c`
  - Custom web application
- `HTTPPrint.h`
  - Custom index file: automatically generated

HTTP2 Process Flow



Dynamic Variables



- Combine system data into web pages
- Present completed page to browser

MICROCHIP
Regional Training Centers

Dynamic Variables

- Indicated in HTML by names in ~~ pairs
- Invoke a callback function
  - Implemented in `CustomHTTPApp.c`
- To create a variable called `foo`:
  - Insert `~foo~` in your web pages
  - Implement `HTTPPrint_foo()`

Definition

Callback Function: A function registered to handle a specific program event. Called automatically by the stack as needed.

Dynamic Variables

Example 1 – Web Page HTML (`index.htm`)

```
<p>System Status: ~status~</p>
```

Example 1 – Callback Function (`CustomHTTPApp.c`)

```
BOOL system_status;

void HTTPPrint_status(void)
{
  if(system_status == TRUE)
    TCPPutROMString(sktHTTP, "on");
  else
    TCPPutROMString(sktHTTP, "off");
}
```

MICROCHIP
Regional Training Centers

Dynamic Variables

- Guarantee 16 bytes free
  - Manage output when writing more
  - Use variable `curHTTP.callbackPos`
    - 0x00000000 on first call
    - Track bytes remaining, or bytes written
    - Invoked again when non-zero on return
    - Restore to zero to indicate completion
  - Ignore for short outputs!

Dynamic Variables

Example 2 – Web Page HTML (`index.htm`)

```
<p>Message: ~long_message~</p>
```

Example 2 –Callback Function (`CustomHTTPApp.c`)

```
BYTE message[150];

void HTTPPrint_long_message(void)
{
  if(curHTTP.callbackPos == 0x00)
    curHTTP.callbackPos = (PTR_BASE)message;
  curHTTP.callbackPos = (PTR_BASE)TCPPutString(sktHTTP,
    (BYTE*)(PTR_BASE)curHTTP.callbackPos);
  if(*(BYTE*)(PTR_BASE)curHTTP.callbackPos == '\0')
    curHTTP.callbackPos = 0x00;
}
```

MICROCHIP
Regional Training Centers

Dynamic Variables

Example 2 – Web Page HTML (`index.htm`)

```
<p>Message: ~long_message~</p>
```

Example 2 –Callback Function (`CustomHTTPApp.c`)

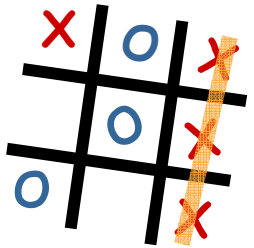
```
BYTE message[50];

void HTTPPrint_long_message(void)
{
  if(TCPIsPutReady(sktHTTP) < 50)
  {
    curHTTP.callbackPos = 0x01;
    return;
  }
  TCPPutString(sktHTTP, message);
  curHTTP.callbackPos = 0x00;
}
```

Microchip Regional Training Centers

Dynamic Variables

- Support parameters
  - `~myVector(3)~, ~myArray(7,25)~`
  - `HTTPPrint_myVector(WORD)`
  - `HTTPPrint_myArray(WORD, WORD)`

Dynamic Variables



Example 4 – Web Page HTML (`tic_tac_toe.htm`)

```
~grid(0,0)~ ~grid(0,1)~ ~grid(0,2)~
~grid(1,0)~ ~grid(1,1)~ ~grid(1,2)~
~grid(2,0)~ ~grid(2,1)~ ~grid(2,2)~
```

Example 4 – Callback Function (`CustomHTTPApp.c`)

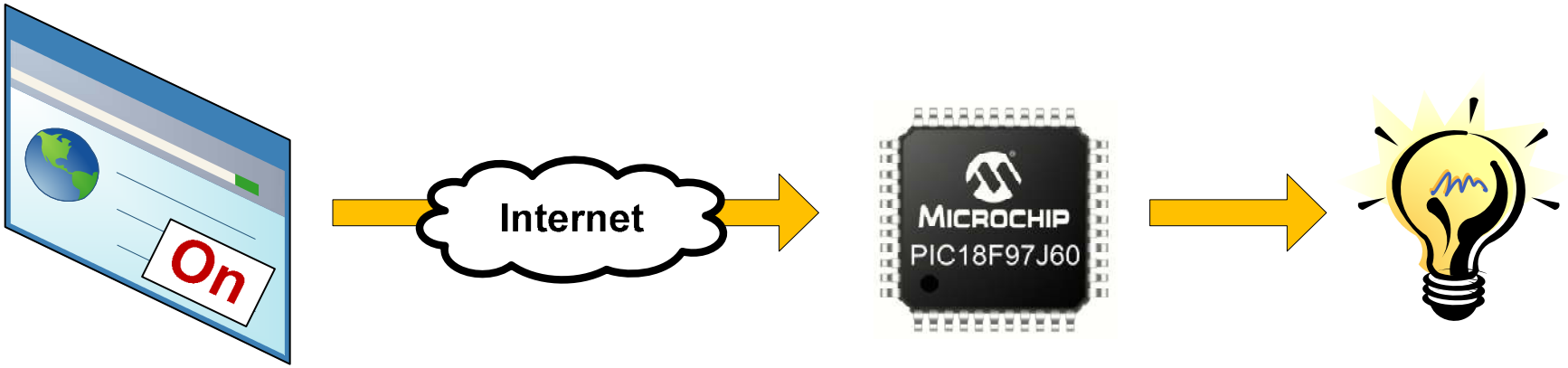
```
BYTE tic_tac_toe[3][3];

void HTTPPrint_grid(WORD row, WORD col)
{
  TCPPut(sktHTTP, grid[row][col]);
}
```

Lab 5: Web-Based Monitoring

- Goals:
  - Replace static text with dynamic variables
  - Display product information
  - Control a bar graph display

- Bonus:
  - Control bar graph colors using CSS classes

Web-Based Control



- Accept input through web pages
  - Manage outputs
  - Control system data

MICROCHIP
Regional Training Centers

Web Form Components

- Designed in HTML
  - Contained within `<form>` tags
- Consists of one or more fields
  - Denoted by `<input name="...">` tags
- Submitted as name/value pairs
  - `lights=on&brightness=50`
  - Non-alphanumeric chars are hex-encoded

Web Form Methods

- **GET** `<form method="get" ...>`
  - Data appended to URL
  - Length limited to ~100 bytes
  - Easiest to process
- **POST** `<form method="post" ...>`
  - Data sent as request body
  - Length is unlimited
  - More difficult to process – advanced class

MICROCHIP
Regional Training Centers

The GET Method

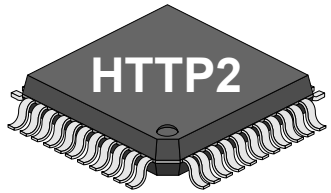
- Data appended to URL
  - `/form.htm?lights=on&brightness=50`
- Easiest to process
  - All input in `curHTTP.data`
  - Automatically decoded
- Limited to available buffer
  - ~100 bytes max input
- Handled in `HTTPExecuteGet()`

The GET Method

```
GET /form.htm?lights=on&brightness=50 HTTP/1.1
Host: 192.168.1.100
...
```

curHTTP.data

```
lights\0on\0brightness\050\0\0
```



HTTPGetArg()

HTTPExecuteGet()

- Locate “lights” parameter
- Control system lights

Microchip Regional Training Centers

GET Method Callback

- Input stored in `curHTTP.data`
- Locate values with:
  - `HTTPGetArg()`
  - `HTTPGetROMArg()`
- Process input values
- Perform necessary actions
- Return:
  - `HTTP_IO_DONE` on completion
  - `HTTP_IO_WAITING` to be called again

GET Method Callback

Syntax

```
BYTE* HTTPGetArg(BYTE* haystack, BYTE* needle)
BYTE* HTTPGetROMArg(BYTE *, ROM BYTE*)
```

- Find parameters with `HTTPGetArg`
- Search for parameters (the needle)
- Search in `curHTTP.data` (the haystack)
- Returns pointer to associated value
  - `NULL` if not found

GET Method Callback

Example 1 – Web Page HTML (`index.htm`)

```
<form method="get" action="index.htm">
  <b>Enter two words below:</b>
  <input type="text" name="size" />
  <input type="text" name="color" />
  <input type="submit" value="Update" />
</form>
```

Example 1 – User Input

```
size: large        color: orange
```

GET Method Callback

Example 1 – `curHTTP.data`

```
size\0large\0color\0orange\0\0
```

Example 1 – `HTTPExecuteGet` Callback (`CustomHTTPApp.c`)

```
BYTE size[20], color[20];
HTTP_IO_RESULT HTTPExecuteGet(void)
{
  BYTE* val;
  val = HTTPGetROMArg(curHTTP.data, (ROM BYTE*)"size");
  if(val)
    memcpy(size, val, sizeof(size) - 1);
  val = HTTPGetROMArg(curHTTP.data, (ROM BYTE*)"color");
  if(val)
    memcpy(color, val, sizeof(color) - 1);
  return HTTP_IO_DONE;
}
```

Lab 6: Control via GET

- Goals:
  - Understand web form
  - Parse input to control LED
- Bonus:
  - Pre-select current state on form

```
GET /lights.htm?lights=off HTTP/1.1
Host: 192.168.2.2
User-Agent: Mozilla/5.0 (Windows; U; Windows NT 5.1; zh-TW; rv:1.9.2.3) Gecko/20100401
Firefox/3.6.3 ( .NET CLR 3.5.30729)
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: zh-tw,en-us;q=0.7,en;q=0.3
Accept-Encoding: gzip,deflate
Accept-Charset: UTF-8,*
Keep-Alive: 115
Connection: keep-alive
Referer: http://192.168.2.2/lights.htm?lights=on
```

Browser

```
HTTP/1.1 200 OK
Connection: close
Content-Type: text/html
Cache-Control: no-cache

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 STRICT//EN" "DTD/xhtml1-strict.dtd">
<html>
<head>
<title>Microchip WebVend Demo App</title>
<link href="/vending.css" rel="stylesheet" type="text/css" />
</head>
<body>
<div id="title"><div class="page">
<div class="right">WebVend Demo App</div>
<img src="/mchp.gif" />
</div><div class="spacer"></div></div>

<div id="content">
<p align="center" style="margin-top: 0">
<a href="index.htm">Status</a>  | 
<b>Lights</b>  | 
<a href="products.htm">Products</a>
```

Explorer 16

MICROCHIP
Regional Training Centers

Review: Building with HTTP2

- Dynamic Variables
  - Generate dynamic pages
  - Insert displayable text or code
- Web-Based Control via GET
  - Easier
  - Limited to 100 bytes
- Advanced Control via POST
  - Unlimited length
  - More complex processing

```
POST /products.htm HTTP/1.1
Host: 192.168.2.2
User-Agent: Mozilla/5.0 (Windows; U; Windows NT 5.1; zh-TW; rv:1.9.2.3) Gecko/20100401
Firefox/3.6.3 ( .NET CLR 3.5.30729)
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: zh-tw,en-us;q=0.7,en;q=0.3
Accept-Encoding: gzip,deflate
Accept-Charset: UTF-8,*
Keep-Alive: 115
Connection: keep-alive
Referer: http://192.168.2.2/products.htm
Content-Type: application/x-www-form-urlencoded
Content-Length: 286

name%5B0%5D=Cola&price%5B0%5D=%242.00&name%5B1%5D=Diet+Cola&price%5B1%5D=%241.00&name%5B2%5D=Root
+Beer&price%5B2%5D=%241.00&name%5B3%5D=Orange&price%5B3%5D=%241.00&name%5B4%5D=Lemonade&price%5B4%
5D=%241.25&name%5B5%5D=Iced+Tea&price%5B5%5D=%241.75&name%5B6%5D=Water&price%5B6%5D=%
242.00HTTP/1.1 200 OK
```

Browser

```
242.00HTTP/1.1 200 OK
Connection: close
Content-Type: text/html
Cache-Control: no-cache

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 STRICT//EN" "DTD/xhtml1-strict.dtd">
<html>
<head>
<title>Microchip Webvend Demo App</title>
<link href="/vending.css" rel="stylesheet" type="text/css" />
</head>
<body>
<div id="title"><div class="page">
<div class="right">Webvend Demo App</div>
<img src="/mchp.gif" />
</div><div class="spacer"></div></div>

<div id="content">
<p align="center" style="margin-top: 0">
<a href="index.htm">Status</a>  | 
<a href="lights.htm">Lights</a>  | 
<b>Products</b>
</p>

<div id="location">Machine WEBVEND          :: Building C4 - 2nd Floor NW</div>

<form method="post" action="products.htm">

<fieldset>
<legend><b>Manage Products</b></legend>

<table align="center" cellpadding="2">
<tr>
<td><b>Display Name</b></td><td><b>Price</b></td>
```

Explorer 16

MICROCHIP
Regional Training Centers

Post Method per Instructor Choice

- Post Method lecture and labs have been moved to the appendix and will be covered in the Advanced HTTP class, COM4302
- If time remains, use Wireshark to show incoming GET form traffic, then trace in the debugger
  - Breakpoint at top of HTTPExecuteGet
  - Watch on curHTTP (look at .data)