MICROCHIP
MASTER'S

643 TCP

Microchip TCP/IP Stack Hands-On

Topics

- What is Microchip TCP/IP Stack ?
- What can I do with it ?
- How do I use it ?
  - Architecture
  - TCP/IP Stack Design
  - Application Interface
- Hands-on

Objectives

- Goal:
  - To explain enough details about Microchip Stack for you to be able to use it.
- Requirements:
  - Should have working knowledge of ‘C’
  - Should have general idea of Internet/Intranet
  - Working knowledge of HTML design

What is Microchip TCP/IP Stack ?

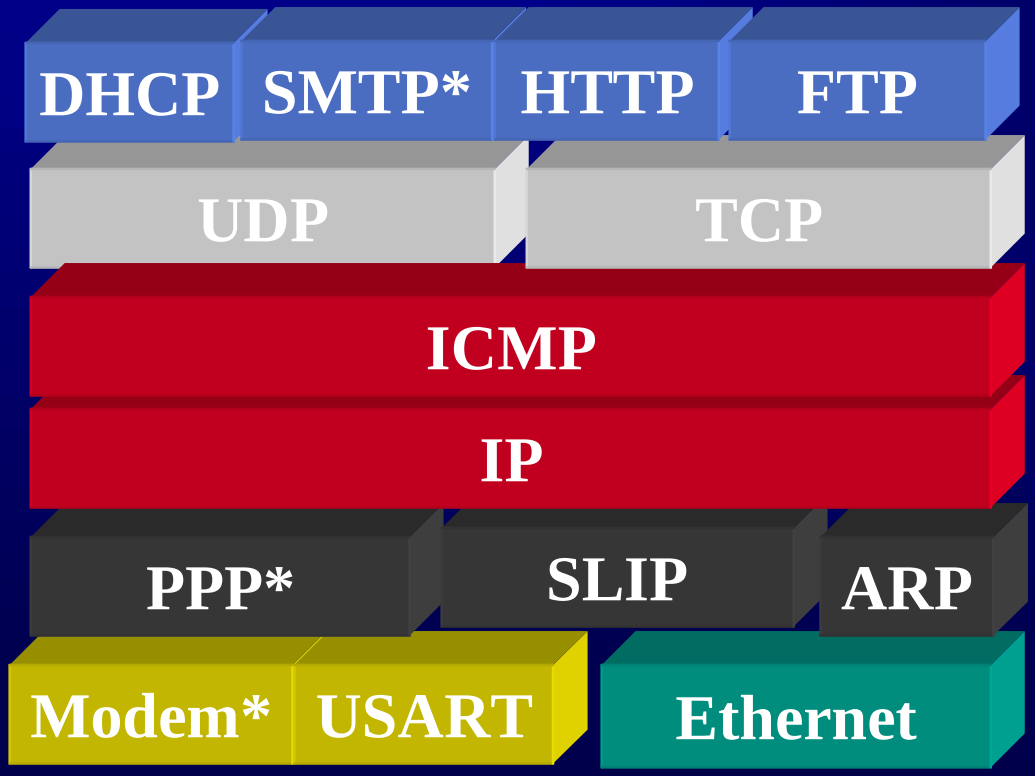
- Microchip App Note AN833
- Source code available for FREE !
  - No royalty or license fee
  - Order #DS39594 - CD ROM (lit_inquiry@microchip.com) - On-line Download in future
- Free technical support
- Suite of files
  - ‘C’ Source files and PC based utility

Main Features

- Portable across PIC18 microcontrollers
- Out-of-box support for Microchip MPLAB® C18 and PICC 18™ (Hitech) compilers
- RTOS independent
- Full TCP State machine
- Modular Design
- Socket support for TCP and UDP

Stack Modules

Internet Protocol Stack



Application Layer

Transport Layer

Internet Layer

Network Access Layer

Physical Layer

* To be added

What can I do with it ?

- Remote control via Web Server
- Remote notification
- Remote upgrade
- Data collecting node for Data Warehousing
- Custom Client/Server application
- Low end Gateway
- Protocol Bridge applications
  - Ethernet to USART, CAN, I2C, SPI etc.

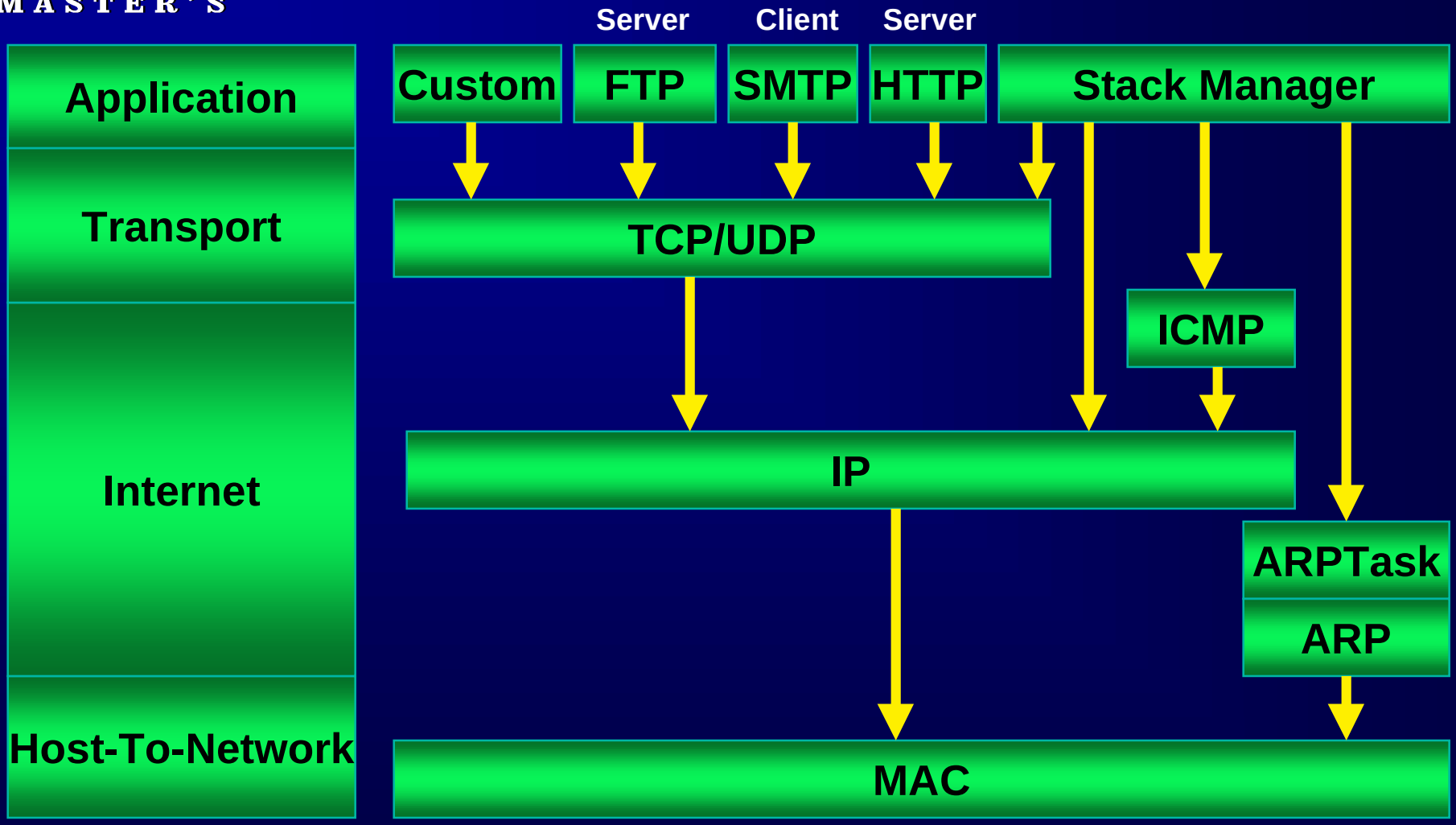
How do I use it ?

- Use PIC18 microcontroller family
- Read AN833 Document
- Sample Application
- Sample projects
- Know your APIs
- No TCP/IP knowledge required for HTTP Server Usage

Architecture

- Multi-tasking co-operative library
  - Main Application must match
- Closely follows TCP/IP Reference Model
- Modular architecture
- Separate Source file for each module
- Compile time configurations

Block Diagram



Microchip Stack Decisions

- No RTOS
  - Uses own multi-tasking approach
- A Library
  - User application has complete control
- Not an interrupt-driven
  - Multi-tasking polling under application control
- Partial TCP Spec. compliance
- Partial HTTP compliance

Stack as a Library

- Problem: User has to know all APIs
  - Difficult to use and maintain
  - User application must manage active modules
- Solution: Provide “Module Manager”
  - Wraps module management
  - Simplifies User application logic
  - Creates high abstraction layer

Microchip Stack Files

Module	File Name
MAC	**MAC.***
ARP	**ARP.***
ARP Manager	**ARPTsk.***
IP	**IP.***
ICMP	**ICMP.***
TCP (+ Manager)	**TCP.***
UDP (+ Manager)	**UDP.***
DHCP (+ Manager)	**DHCP.***
HTTP Server	**HTTP.***
SMTP Client	**SMTP.***
Stack Manager	**StackTsk.c**

- Not all files are not listed

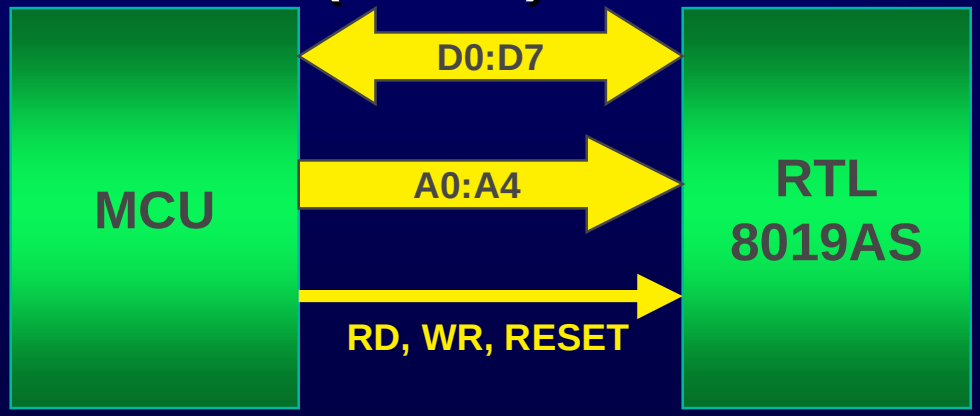
Microchip Stack Samples

- Multiple MPLAB Projects to demonstrate different Configurations (Main: websrvr.c)
- Uses same Source files - Different options

Project	Purpose
HtNICEE.pjt	Hi-Tech, NIC, MPFS in external EEPROM
HtNICPG.pjt	Hi-Tech, NIC, MPFS in Program Memory
HtSlEE.pjt	Hi-Tech, SLIP, MPFS in external EEPROM
HtSlPG.pjt	Hi-Tech, SLIP, MPFS in Program Memory
MpNICEE.pjt	C18, NIC, MPFS in external EEPROM
MpNICPG.pjt	C18, NIC, MPFS in Program Memory
MpSlEE.pjt	C18, SLIP, MPFS in external EEPROM
MpSlPG.pjt	C18, SLIP, MPFS in Program Memory

Media Access Control (MAC)

- Out-of-box support for RealTek 8019AS
  - Easy port to other NE2000 NICs
- Uses PICDEM.net™ Internet board as HW platform
  - Easily modifiable for other boards
- Uses NIC RAM (8 KB) as buffer



Microchip Stack Socket

- One of many channels for a connection
- Allows multiple concurrent connections
  - `'N'` simultaneous HTTP Connections
  - `'M'` simultaneous TCP, UDP applications
- All sockets share reusable buffers
  - PRO: Less memory per socket
  - CON: Socket must consume its buffer in one task time
- Static allocation

Transmission Control Protocol (TCP)

- Connection Oriented - Reliable data transfer
- Full TCP State Machine
  - So is the TCP module footprint
- Unlimited TCP Sockets
  - Limited by available RAM and compiler only
- Automatic packet fragmentation
- “TCP Window” of one segment
  - Can be configured to not use Tx Window
- In-place TCP checksum calculation

User Datagram Protocol (UDP)

- Based on IP
- Connectionless Protocol
- No built-in error recovery
- Unlimited UDP Socket support
  - Socket count limited by available RAM and compiler only
  - Multiple applications can access UDP simultaneously
- No checksum calculation

IP Configuration

- IP Address, Subnet Mask, Gateway
- Static values
  - Hard-coded in source code
- IP Gleaning
  - Only IP address
- Dynamic Host Configuration Protocol
  - All IP parameters
  - Fully automatic

IP Gleaning

- Simple, lean method
  - Uses ARP and ICMP
- Configures IP Address only
- Steps:
  - 1. Modify host's ARP cache
  - 2. Ping to remote node with desired IP address
- Handled automatically

Dynamic Host Configuration Protocol (DHCP)

- Fully automatic method
  - Node gets configured on power-up
  - Automatic IP renewal
- Needs at least one DHCP Server on network
- Configures
  - IP, Gateway Address, Subnet Mask
- Embedded Systems Problem:
  - Discovery of IP Address
- Not available with SLIP module

HTTP Server

- Multiple simultaneous connections
- Supports HTML Forms
- Dynamic web page creation
- Pages stored in Program Memory or external data EEPROM
- Simple and powerful Microchip File System (MPFS)
- Easy to integrate

Microchip File System (MPFS)

- Small yet powerful file system
- Flexible storage scheme
  - Internal program memory or external data EEPROM (up to 64KB)
- PC based utility to generate MPFS image
- 8 + 3 Short file names
- Case-insensitive file names
- Read AN833 for more detail

MPFS Image

- Two types
  - ‘C’ data file for Program Memory
  - ‘bin’ file for external data EEPROM
- PC utility “mpfs.exe”
- All web pages in one directory
- Image size must fit in available memory
- “CR LF” stripped from “*.htm” files
- Reserved block for app. specific data

MPFS Utility

- `mpfs [/?] [/c] [/b] [/r<Block>] <InputDir> <OutputFile>`
  - `/?` : Display help
  - `/c` : Generate ‘C’ data file
  - `/b` : Generate binary data file (Default)
  - `/r` : Reserve a block of memory at beginning (Used in /b mode only. Default=32)
  - `<InputDir>`: Directory that contains files
  - `<OutputFile>`: Output file name

MPFS Examples

- Generate ‘C’ data file
  - `mpfs /c MyPagesDir mypages.c`
  - Link this file into your project
- Generate binary data file, with 32 bytes reserved block
  - `mpfs MyPagesDir mypages.bin`
- Reserve 128 bytes in EEPROM
  - `mpfs /r128 MyPagesDir mypages.bin`
  - Useful to store custom data outside MPFS

HTTP Server API

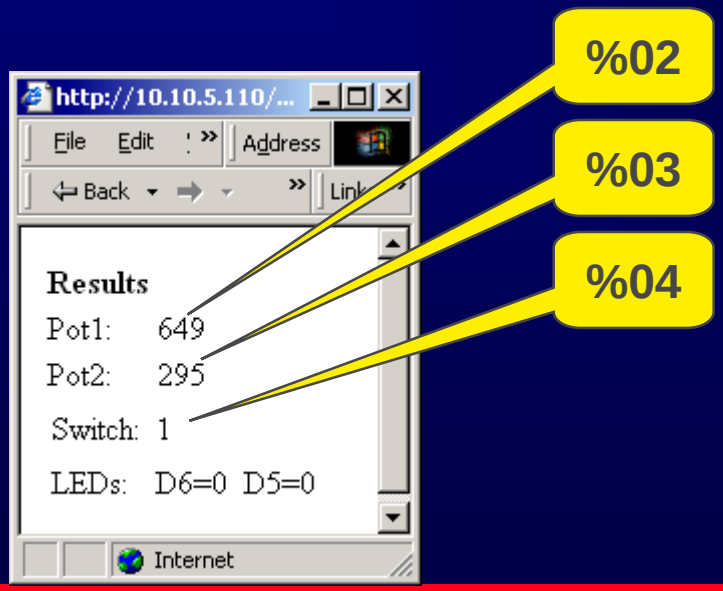
- **HTTPInit**
- **HTTPServer**
  - Performs HTTP Server tasks - Call periodically
- **HTTPGetVar**
  - Callback to get dynamic variable value
- **HTTPExecCmd**
  - Callback to execute HTML form command

Dynamic HTML Pages

- Must have “cgi” file extension
- Variable substitution method
- Format: **%xx - xx** is a variable (0-99)
- Substitution may be one or more characters
  - May be used to upload complete binary image
- Use extra ‘%’ to display % itself
  - **23%% displays 23%**

Variable Substitution Example

```
1. <table>
2. <tr><td><b><Results></b></td></tr>
3. <tr><td><Pot1:></td><td>%02</td></tr>
4. <tr><td><Pot2:></td><td>%03</td></tr>
5. <tr><td><Switch:></td><td>%04</td></tr>
...
```



Dynamic HTML Pages - Uses

- Change web page content
  - ...Serial Number=%01…
- Change graphics based on a variable
  - img src=LED%02.gif
- Change page link
  - LinkName
- Change client-side scripts (java etc.)

HTTPGetVar

- **(BYTE *var*, WORD *ref*, BYTE **val*)**
- Data transferred byte at a time
- ***ref*** is used for multi-byte transfers
  - First transfer with ***ref*** = **HTTP_START_OF_VAR**
  - return other than **HTTP_END_OF_VAR** to indicate multi-byte transfer
  - Finish data transfer by returning **HTTP_END_OF_VAR**

HTTPGetVar Example

- (BYTE *var*, WORD *ref*, BYTE **val*)

```
1. if (var == 4) // Identify variable.
2. { // Return ‘1’ if switch is open, else ‘0’
3.    if ( RB5 ) *val = ‘1’;
4.    else *val = ‘0’;
5.    return HTTP_END_OF_VAR;
6. }
7. else
8. // Check for other variable...
...
```

Multi-byte transfer

- (BYTE *var*, WORD *ref*, BYTE **val*)

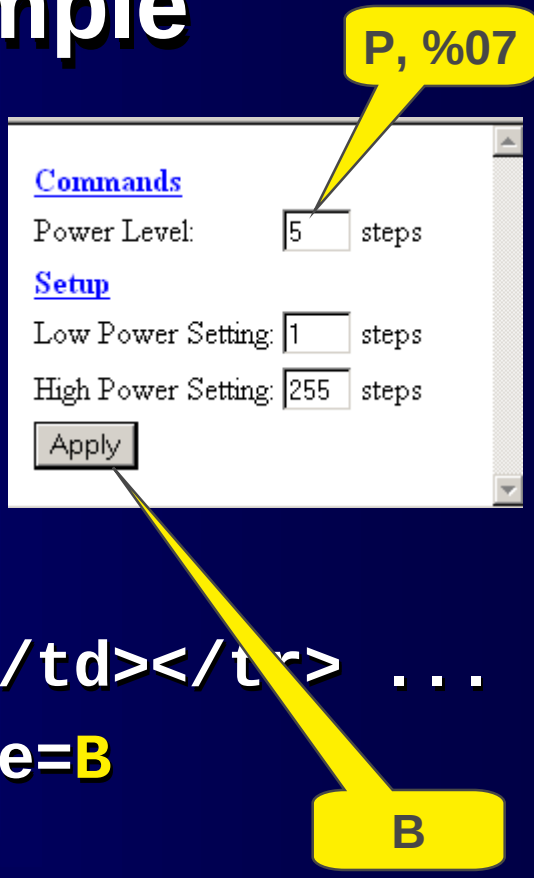
```
1. if (var == 1)  // Identify var.
2. { // If this first call, init array index.
3.    if ( ref == HTTP_START_OF_VAR ) ref = 0;
4.    // Stuff current byte in buffer.
5.    *val = SerialNumberStr[(BYTE)ref];
6.    if ( *val ==‘\0’ ) return HTTP_END_OF_VAR;
7.    return ++ref;  // Advance array index
8. }
9. else // Check for other variable...
...
```

HTML Forms

- Interactive HTML pages
  - Data is transferred from PC to PIC18 microcontroller
- Form method '**GET**' only
- Remote command invocation
  - User application must implement the command
- Caution: Multiple users may execute same command "simultaneously"
  - Must protect critical data

Form Example

```
1. <FORM METHOD=GET
            action=command.cgi>
2. <table>
3. <tr><td>Power Level:</td>
4. <td><input type=text size=2
   maxlength=1 name=P value=%07></td></tr> ...
5. <tr><td><input type=submit name=B
   value=Apply></td></tr> ...
```



- Action by browser when *Apply* is clicked

```
10.10.5.110/command.cgi?P=5&L=1&H=255&B=Apply
```

HTTPExecCmd

- **(BYTE **argv, BYTE argc)**
- **argv[0]** = Form Action Name
- **argv[1…argc]** = Parameter **1** to **argc**
- **argc** = Number of parameters including Form Action Name
- **command.cgi?P=5&L=1&H=255&B=Apply**
  - argv[0] = “command.cgi”, argv[1] = “P”, argv[2]= “5” etc… ; argc = 9
  - argv[0] determines page to be uploaded next

HTTPExecCmd Example

- (BYTE **argv, BYTE argc)

```
1. for ( i = 1; i < argc; i++ ) {
2.     if ( argv[i][0] == 'P' ) // Power setpoint ?
3.           PowerVal = atoi(argv[++i]); // Save value
4.     else if ( argv[i][0] == 'L' ) // Low setting ?
5.           LowPowerSetting = atoi(argv[++i]); // Save
6.     else if ( argv[i][0] == 'H' ) // High setting ?
7.           HighPowerSetting = atoi(argv[++i]); // Save
    }
}
// If required, another page may sent as a result
strcpy(argv[0], "RESULTS.CGI"); // Set result page
```

Form Fields Limit

- Allowable max. number of arguments
  - FormAction?Arg1=Val1+Arg2=Val2…
  - See **`MAX_HTTP_ARGS`** & **`MAX_HTML_CMD_LEN`** in "**`http.c`**"
- Default
  - **`MAX_HTTP_ARGS = 5 (Includes action)`**
  - **`MAX_HTML_CMD_LEN = 80`**
- If limit is exceeded,
  - Extra arguments are ignored

Web Page Types

- Page file extension defines how browser displays/interprets the page
  - Default support for “txt”, “htm”, “gif”, “cgi”, “jpg”, “cla”, “wav”.
  - If needed, modify “httpFiles” and “httpContents” in http.c file
- “index.htm” is the default web page
  - Defined by HTTP_DEFAULT_FILE_STRING in http.c file

HTTP Server Integration

- Integrate Stack files in your project
  - See sample project in AN833
- Implement HTTP Server Callbacks
- Design your web-site
  - Must contain “index.htm” - Default web page.
- Create MPFS image
- Program your MPFS storage

Main Application Integration Example

```
1.  void main(void) {
2.    // Initialize hardware
3.    TickInit(); MPFSInit(); StackInit();
4.    HTTPInit();     // + Other stack app init
5.    while(1){ // Main infinite loop
6.          StackTask();
7.          HTTPServer(); // + other stack tasks
8.          // Your app specific task(s)
9.    } }
10. ISR() {     // Interrupt Handler
11.   TickUpdate(); } + HTTP Callbacks
```

Web Page Design Guidance

- Avoid using excessive files in a page
  - More simultaneous connections
  - More RAM usage
- Hand-code the pages
  - Or remove unnecessary tags generated by visual web authoring tool
- Try to shrink the graphics
  - Use correct file formats
- Auto refresh dynamic content only

Security

- No built-in security
- App. must provide reasonable authentication and encryption
- Avoid blind remote control
- Restrict critical commands
- PIC® microcontroller is not same as PC
  - Limit no. of users to designed limit
  - No built-in solution for “Denial Of Service”

Microchip Stack Configuration

- Many compile-time options
  - Default IP configuration
  - Module Selection
  - Socket Counts
  - Buffer size
  - MPFS storage type etc.
- Automatic compile-time validation
  - Automatic module enable/disable
- It's all in "**`StackTsk.h`**" file

Configuration Parameters - I

1	**CLOCK_FREQ**	Oscillator Frequency
2	**TICKS_PER_SECONDS**	Timebase for timeouts
3	**TICK_PRESCALE_VALUE**	Timer prescale value
4	**MPFS_USE_PGRM**	Use Program memory
5	**MPFS_USE_EEPROM**	Use EEPROM
6	**MPFS_RESERVE_BLOCK**	App. Specific block
7	**STACK_USE_ICMP**	Enable ICMP module
8	**STACK_USE_SLIP**	Enable SLIP module
9	**STACK_USE_UDP**	Enable UDP module
10	**STACK_USE_IP_GLEANING**	Enable IP Gleaning module
11	**STACK_USE_DHCP**	Enable DHCP module
12	**STACK_USE_TCP**	Enable TCP module
13	**MY_DEFAULT_???**	Default IP Configurations

Configuration Parameters - II

14	**MAX_SOCKETS**	Maximum TCP Sockets
15	**MAC_TX_BUFFER_SIZE**	MAC transmit buffer size
16	**MAC_TX_BUFFER_COUNT**	Total MAC transmit buffers
17	**MAX_HTTP_CONNECTIONS**	Maximum HTTP connections
18	**TCP_NO_WAIT_FOR_ACK**	Disable tx TCP window
19	**STACK_CLIENT_MODE**	Enable Client APIs
20	**MAX_UDP_SOCKETS**	Maximum UDP Sockets

- Always check "**`StackTsk.h`**" for up-to-date list
- Also check "**`version.log`**" for version history/changes

Server Or Client ?

- Two modes of operations
  - Server
  - Server and Client
- **`STACK_CLIENT_MODE`** enables Client mode
- Client mode provides
  - **`TCPConnect`**, **`ARPResolve`**, **`ARPIsResolved`**
- SMTP module will enable Client mode

Hands-On

- Requirements:
  - Working knowledge of ‘C’
  - HTML knowledge useful
  - PICDEM.net Internet demo board
  - MPLAB ICD-2 or ICE2000
  - Microchip MPLAB C18 compiler
  - PC with Ethernet card
- Use `c:\Stack\mpnicee.pjt` project

Hands-On 1

- Goal: Learn how to modify MPFS image
- Exercise:
  - Modify c:\Stack\WebPages\Main.htm
  - Build MPFS image using c:\Stack\mpfs.exe
  - Download image
  - Use AN833 for reference

Hands-On 2

- Goal: Learn about command execution via browser
- Exercise:
  - Modify `c:\Stack\Websrvr.c` file
  - Implement `HTTPExecCmd()` callback to toggle LEDs D5, D6 when “Toggle D5” and “Toggle D6” buttons are pressed respectively.

Hands-On 3

- Goal: Learn how to dynamically change web pages
- Exercise:
  - Display graphical LED state - LED1.gif for ON and LED0.gif for OFF.

Hands-On 4

- Goal: Learn about PC to PIC data transfer using forms
- Exercise:
  - Implement code to accept “User Name” from *Output Control* and save it to PIC microcontroller RAM.

Hands-On 5

- Goal: Learn about PIC microcontroller to PC data transfer
- Exercise:
  - Display saved “User Name” in *System Status* page.