



Microchip TCP/IP Stack

Topics

- Main Features
- What can I do with it ?
- Web Server & Microchip File System
- Web Server Integration
- Web page design and security

Requirements

- Working knowledge of ‘C’
 - Some knowledge of “C pointers” useful
- Working knowledge of HTML page design
- No TCP/IP knowledge required for Web Server application
- Familiarity with co-operative multi-tasking coding technique useful

Main Features

- Portable across PIC18 microcontrollers
- Out-of-box support for Microchip C18 and Hi-Tech PICC-18 compilers
- RTOS independent
- Full TCP State machine
- Modular Design
- Socket support for TCP and UDP

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.

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
HtSIEE.pjt	Hi-Tech, SLIP, MPFS in external EEPROM
HtSIPG.pjt	Hi-Tech, SLIP, MPFS in Program Memory
MpNICEE.pjt	C18, NIC, MPFS in external EEPROM
MpNICPG.pjt	C18, NIC, MPFS in Program Memory
MpSIEE.pjt	C18, SLIP, MPFS in external EEPROM
MpSIPG.pjt	C18, SLIP, MPFS in Program Memory

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

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><Results></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 on-the-fly
 - ...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 PIC
- 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

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 be 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 projects 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 is not same as PC
 - Limit no. of users to designed limit
 - No built-in solution for “Denial Of Service”