

# ColdFire Ethernet for Diverse Applications

## Including an RSS/XML Feed Reader

by: Eric Gregori  
Product Specialist — Embedded Firmware  
Sales and Marketing

### 1 Embedded Ethernet

Ethernet has become a reality for low-cost embedded systems. The Ethernet standard (IEEE 802.3) was originally designed for networking computers over local area networks (LAN), but it has since been adapted for other purposes. Today it has become so popular that it is hard to find a PC or laptop without an Ethernet port. Now this capability is migrating to the embedded world, where the ColdFire® family excels. The IEEE 802.3 specification defines a mechanical/electrical connection between devices (physical layer) and a multi-node addressable communications protocol (Media Access Control — MAC — layer).

#### 1.1 Ethernet Physical Layer

The Ethernet physical layer defines the physical connections between nodes. The 802.3 standard defines many physical layers, including everything from coaxial cable to fiber optics. Through the years, the most common choice for this purpose has changed drastically from thick multi-strand cables with large connectors (called thicknet) to the small RJ-45 8-pin connector we use today.

The common modern copper physical layer is referred to as 100Base-TX (a type of 100Base-T). This copper-based twisted-pair medium contains eight wires grouped in four twisted pairs. Two twisted pairs are used for communication in each direction. The cable is referred to as a category 5 (cat 5 for short). The category 5 standard defines a cable consisting of four twisted pairs capable of carrying frequencies up to 100 MHz.

Most Ethernet embedded devices have integrated MACs (discussed in the next section) with external PHY. ColdFire offers a seven-member family of microcontroller units (MCU) with integrated MAC/PHY called MCF5223x.



Figure 1. The Modern Ethernet Jack

Most modern buildings and residences are wired using category 5 cables for their PC networks and broadband, making this an ideal medium for distributed processing/sensing in a building or residential environment.

## 1.2 Ethernet MAC Layer

The MAC protocol layer defines the communication that occurs over the physical layer. Ethernet is a multi-node protocol, so each node has a unique address. This address is defined in the MAC layer. In Ethernet communication, MAC addresses are 48 bits long (six bytes, or octets). The device’s MAC address never changes — usually it is programmed at the factory. The MAC address must be unique, so MAC addresses are managed and distributed by the IEEE.

The MAC address, along with various other fields, is contained in the Ethernet MAC header. As the name implies, the header sits in front of the Ethernet packet. It contains the MAC address of the source node, along with the MAC address of the destination node, and a type field.

Preamble	Destination Address	Source Address	Frame Type	Frame User Data	FCS Checksum
8 Bytes	6 Bytes	6 Bytes	2 Bytes	46 – 1500 Byte	4 Bytes

Figure 2. Ethernet MAC Header

Ethernet can be used directly without any additional layers. It provides a simple point-to-point communication mechanism, with some error checking (FCS checksum). Ethernet by itself does not provide the high degree of communications robustness we have become accustomed to. Additional layers are required to add features such as multiple ports, packet re-transmission, packet timeouts, and connections. These additional layers are defined by the seven-layer OSI model.

## 1.3 Seven-Layer OSI Model

The seven-layer OSI model defines the functions of the various layers in a communication stack. The lowest layers (physical and MAC/data link layers) are traditionally implemented in hardware. The five layers above the MAC/DDDL are usually implemented in software.

The network or IP layer (for a TCP/IP stack) provides an additional layer of addressing (IP addresses in the hexadecimal format of xx.xx.xx.xx) and multiplexing. Multiplexing splits a single communications channel into multiple time-divided communications channels (ports, in TCP/IP terminology).

The transport layer adds the most critical feature to the communication stack. TCP (transmission control protocol) is one of the transport layers in TCP/IP. This layer is responsible for creating a virtual connection between two logical points (not nodes). The logical points are referred to as sockets. The socket's API is actually defined by the session layer.

Last, at the highest level, is the application layer. This application defines the common protocols used on the Internet: HTTP, SMTP, and TFTP. This layer can also be used for custom protocols.

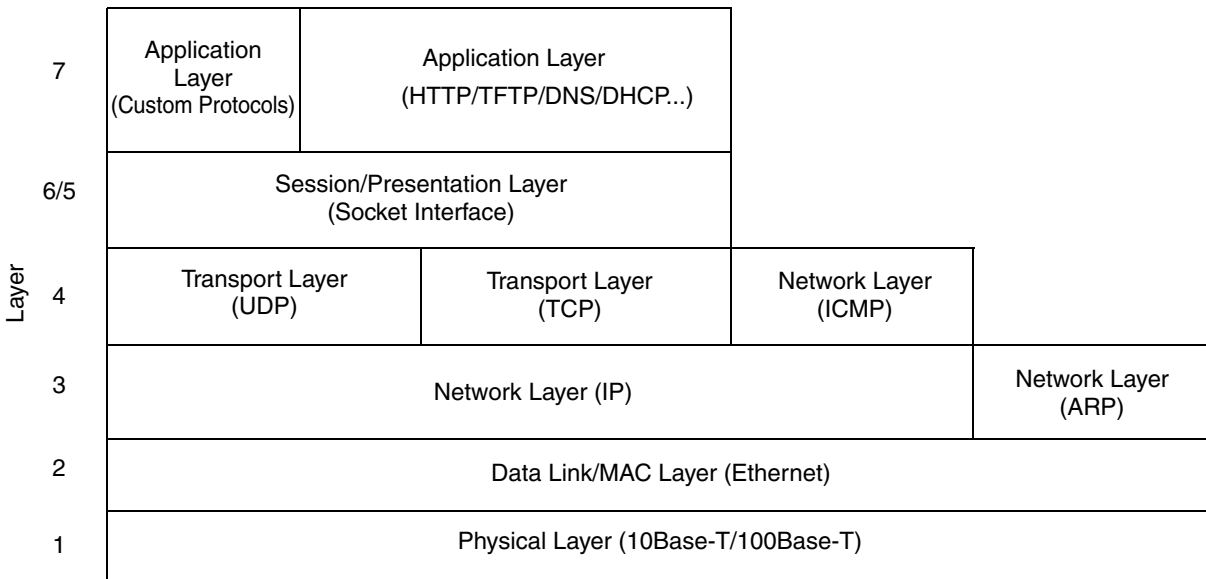


Figure 3. Seven-Layer OSI Model

## 2 ColdFire Family of Microcontrollers

The ColdFire family of microcontrollers is based on the 32-bit ColdFire cores. The ColdFire core is available in four varieties, each a superset of the core below it. The cores are completely scalable, with the differences consisting of additional instructions or add-on modules (for instance, MMU) and longer pipelines to increase frequency and performance for demanding applications. The current V1 core contains the base register and instruction set. The V2 core adds additional instructions and addressing modes to the V1 core, along with an optional eMAC (Enhanced Multiply/Accumulate unit).

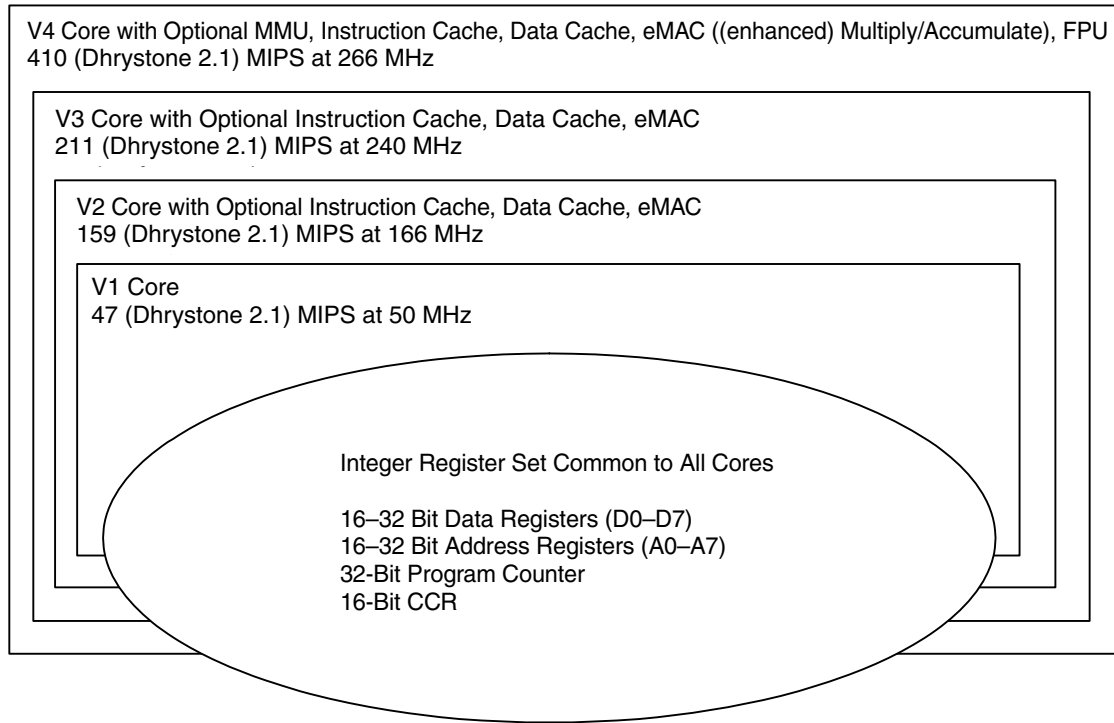


Figure 4. ColdFire Cores: Scalable Instruction Sets, Features, and Performance

## 2.1 Advantage of a 32-Bit Architecture

The true 32-bit architecture of ColdFire microcontrollers lends itself well to efficient communication stack data movement. In a communication stack such as TCP/IP, the packet comes in at the bottom of the stack and propagates up. Data to be sent starts at the top of the stack as a buffer, then works its way to the bottom of the stack to be sent out as a packet.

Movement up and down the stack is an area of inherent inefficiency in a communication stack. To improve efficiency, higher-performance stacks use pointers instead of copying the data multiple times (this is sometimes referred to as zero copy). Pointer arithmetic is significantly more efficient with a 32-bit core using true 32-bit registers. This is a big advantage for the ColdFire 32-bit architecture.

In addition, extracting data from individual fields in a header can become a single-instruction operation by using advanced addressing modes with offset capability.

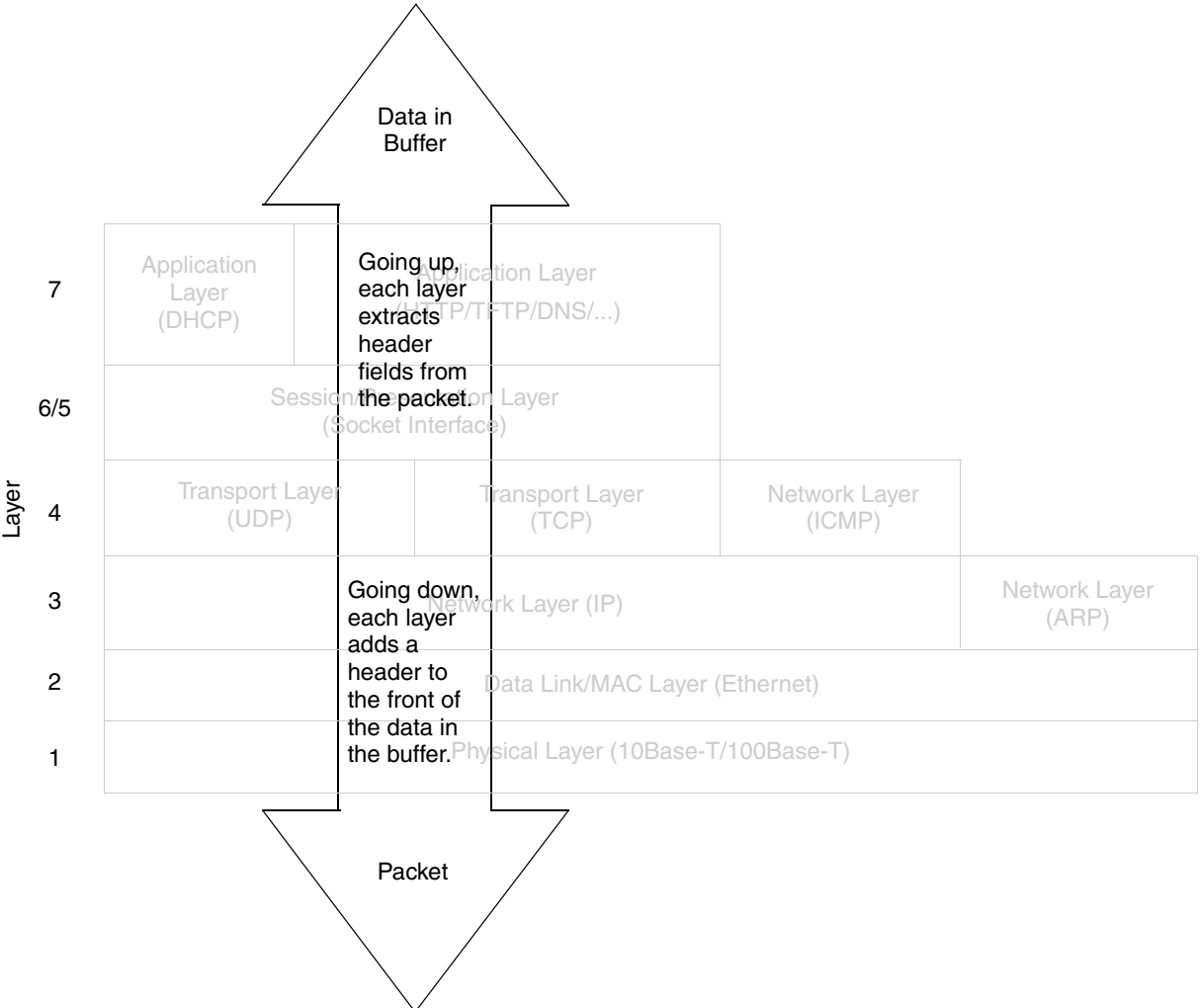


Figure 5. Seven-Layer OSI Model With Communication Stack Overlaid on Top

### 3 ColdFire Fast Ethernet Controller (FEC)

The FEC module is the ColdFire interface to the Ethernet world. The FEC module is consistent from the highest-performance V4-core-based part all the way down to the V1 core. This consistency means that drivers written for one Ethernet-enabled ColdFire processor will work on any Ethernet-enabled ColdFire processor (memory allocation would be the biggest difference).

The FEC module is a high-performance Ethernet engine with a very rich heritage. The FEC module started out in the MPC860T. This high-performance, Power-Architecture-based processor quickly became a powerhouse in the Ethernet world, going into high-performance routers and telecommunication equipment. The MPC860T was so popular in the Ethernet world that if you make a call today, chances are that somewhere along the way the voice data of your call will pass through an MPC860T.

The MPC860T came out in the mid-1990s. The FEC module has been tested and improved upon for over 10 years in some of the highest performance Ethernet environments. The FEC module from the MPC860T is now in the ColdFire line of processors.

### 3.1 ColdFire FEC Features

- Ethernet Media Access Controller (MAC) is designed to support 10 Mbps and 100 Mbps Ethernet/IEEE 802.3 networks
- IEEE 802.3 full-duplex flow control
- Support for full-duplex operation (200 Mbps throughput)
- Retransmission from transmit FIFO after a collision (no use of processor bus)
- Automatic internal flushing of the receive FIFO for runts (collision fragments) and address recognition rejects (no use of processor bus)
- Address recognition
  - Frames with broadcast address may be always accepted or always rejected
  - Exact match for single 48-bit individual (unicast) address
  - Hash (64-bit hash) check of individual (unicast) addresses
  - Hash (64-bit hash) check of group (multicast) addresses
  - Promiscuous mode
- Dedicated DMA controller to allow for packet transmission and reception with no processor overhead.

The Fast Ethernet controller supports both 10 Mbps and 100 Mbps, allowing ColdFire to interface with both old (10Base-T networks) and newer (100Base-TX) networks (with the appropriate 10/100 PHY). Full-duplex operation means that packets can be sent and received at the same time (remember the PHY uses separate wires for TX and RX). At 100 Mbps this translates to a maximum 200 Mbps throughput.

The hardware performs all the functions of the 802.3 Ethernet MAC layer without software intervention. The software simply initializes the FEC, writes the node's MAC address into the MAC address register, and initializes the RX and TX buffer rings. The FEC will automatically receive, process, and verify (via CRC) incoming packets, and DMA the packet into an RX buffer. For TX, the FEC is triggered by software, then automatically DMA's the packet from the TX buffer, calculates a CRC, serializes the packet, and sends it out to the PHY. If a collision is detected, the FEC will perform a random back-off and retry without processor intervention. After the packet is transmitted, the FEC reports a status.

The advantage of this high level of functional integration into the FEC is reduced software overhead. The software simply has to create the packet and give it to the FEC for transmission via the TX ring buffer. On the receive side, the software simply has to take the packet from the RX ring buffer. The ring buffers are managed by the FEC hardware.

## 4 ColdFire TCP/IP Stack

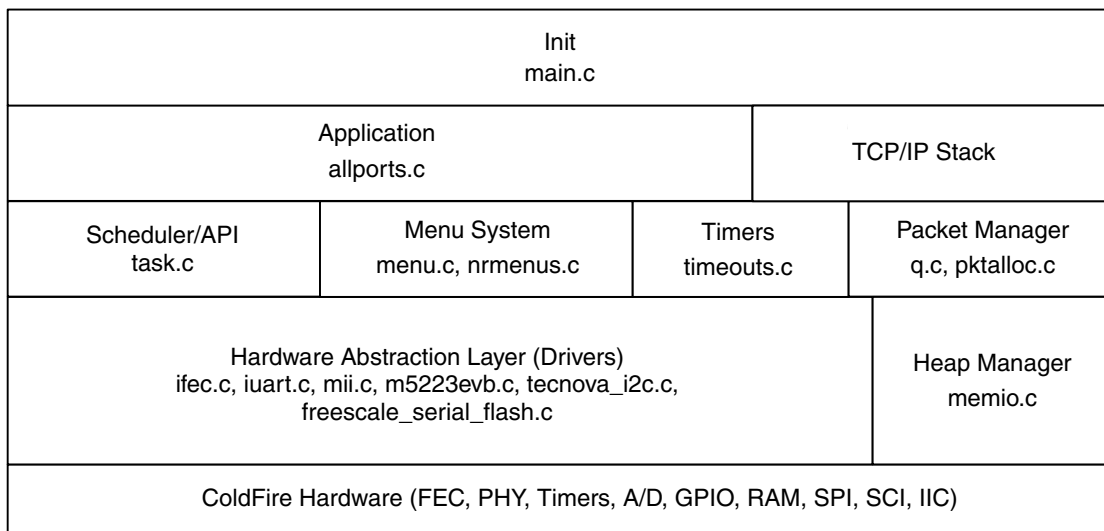
TCP/IP (Transmission Control Protocol/Internet Protocol) is the communication protocol of the Internet. The name refers to two layers of the communication stack: TCP and IP. The term TCP/IP actually

describes multiple protocols within the two layers. Each protocol is defined by an IEEE RFC (request for comment).

Proper TCP/IP stack operation requires multitasking. The ColdFire TCP/IP stack is integrated with a simple multi-tasking operating system. This simple round-robin OS can also be used by the application code. The OS is non-preemptive, but does provide two modes of operation (single-stack (super-loop) and multi-stack). Additional OS features include an interactive real-time upgradeable menu system, user timers, and heap memory management.

## 4.1 ColdFire TCP/UDP/IP Stack Features

- HTTP (Hypertext Transfer Protocol), Serial-to-Ethernet Gateway, TFTP (Trivial File Transfer Protocol)
- Mini-IP Application Interface
- DHCP (Dynamic Host Configuration Protocol) or manual IP configuration, DNS (Domain Name System)
- TCP (Transmission Control Protocol), UDP (User Datagram Protocol)
- ICMP (Internet Control Messaging Protocol), BOOTP (BOOTstrap Protocol)
- ARP (Address Resolution Protocol), IP (Internet Protocol)



**Figure 6. ColdFire TCP/IP Stack and RTOS**

The TCP/IP stack implements the protocols described in these RFCs (please refer to [www.rfc-editor.org/rfcxx00.html](http://www.rfc-editor.org/rfcxx00.html) for details):

- RFC 791: *Internet Protocol (IP)*
- RFC 792: *Internet Control Message Protocol (ICMP)*
- RFC 768: *User Datagram Protocol (UDP)*
- RFC 793: *Transmission Control Protocol (TCP)*
- RFC 826: *Ethernet Address Resolution Protocol (ARP)*
- RFC 1035: *Domain Names - Implementation and Specification (DNS)*

## ColdFire TCP/IP Stack

- RFC 2131: *Dynamic Host Configuration Protocol*
- RFC 2132: *DHCP Options and BOOTP Vendor Extensions*

The session/presentation layer is a mini-socket interface similar to the familiar BSD socket interface. The stack has been optimized for embedded applications using zero-copy functionality for minimum RAM usage.

Freescale Web Server	Freescale Compile-Time Flash file	Freescale Run-Time Flash file	
ColdFire_TCP/IP_Lite—RTOS and Console			
ColdFire_TCP/IP_Lite—Mini-Socket TCP API			
ColdFire_TCP/IP_Lite—TCP	ColdFire_TCP/IP_Lite—UDP	ColdFire_TCP/IP_Lite—ICMP	
ColdFire_TCP/IP_Lite—IP Layer			
ColdFire_TCP/IP_Lite—FEC Driver			
Freescale Ethernet PHY	Freescale Hardware API		

## 4.2 DHCP Client

The Dynamic Host Configuration Protocol is used to acquire network parameters at runtime. The DHCP protocol is defined in RFC 2131 and RFC 2132. The stack runs a DHCP client which searches for a DHCP server (this is referred to as discovery).

Packets are transferred using the UDP layer and BOOTP ports (67 and 68). Because the IP stack does not have an IP address yet, discovery is done using strictly broadcast addresses. Included in the discovery packet is a unique transaction ID (xid). A listening DHCP server sends an offer message containing the xid sent by the client and the suggested network parameters, again using broadcast addressing. Also encoded in the offer is a unique server ID. The client will use this server ID when sending a request packet back to the server, indicating that it accepts the network parameters that were offered. Finally the server ACK's the client using its new IP address.

## 4.3 DNS Client

The DNS client is used to communicate with the DNS (Domain Name Server). The purpose of the DNS system is to translate domain names into IP addresses. The DNS protocol is described in RFC 1035. DNS can use UDP or TCP, with port 53. The DNS protocol is stateless — all the information is contained in a single message. This message is fully documented in RFC 1035.

## 5 Available Examples and Application Notes

All application notes mentioned in this document are available at [www.freescale.com](http://www.freescale.com).

### 5.1 HTTP Web Server and Flash File System

The HTTP web server and flash file system are described in detail in application note AN3455, *ColdFire Lite HTTP Server*.

The features are:

- HTTP 1.0 compliant server with connection persistence and multiple sessions
- Multiple HTTP connections supported
- Flash file system which supports both ColdFire internal flash and external SPI flash
- Web pages can be updated in flash over Ethernet or built in at compile time
- HTTP GET method supported, with a simple mechanism for adding other methods
- Dynamic HTML (Hypertext Markup Language) support with replace and conditional tokens
- Serial interface support for Dynamic HTML variables
- Run-time and compile-time flash file systems
- Long filename support with subdirectories
- “DIR” command supported on serial interface
- PC utilities for compressing compile-time and run-time downloadable images of multi-page web pages
- PC utility for downloading run-time downloadable web page image through port 80 (to get through firewalls)
- 32-byte ASCII key for web page download security

### 5.2 UDP/TCP Clients and Servers — Example Source Code

The ColdFire Lite stack project includes almost a dozen built-in usage examples. These examples are designed to highlight various features in the stack and demonstrate how to use them. The TCP/IP stack and RTOS, along with all the sample applications listed below, are discussed in AN3470, *ColdFire TCP/UDP/IP Stack and RTOS*.

Code examples include:

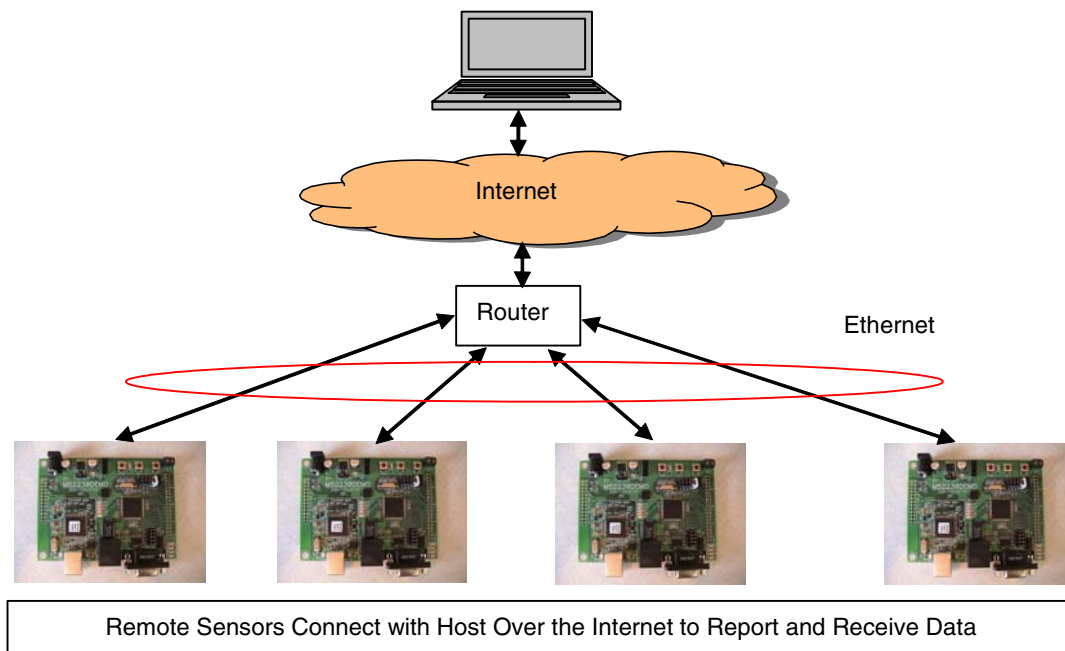
- ColdFire\_Lite  
Barebones TCP/IP stack
- ColdFire\_Lite\_RTOS  
How to use the RTOS application
- ColdFire\_Lite\_TFTP  
TFTP server application
- ColdFire\_Lite\_UDP\_client  
UDP client application for UDP performance testing

## Available Examples and Application Notes

- ColdFire\_Lite\_UDP\_server  
UDP server application for UDP performance testing
- ColdFire\_Lite\_TCP\_client  
TCP client application for TCP performance testing
- ColdFire\_Lite\_TCP\_server  
TCP server application for TCP performance testing
- ColdFire\_Lite\_TCP\_serial\_client  
TCP to serial/serial to TCP client
- ColdFire\_Lite\_TCP\_serial\_server  
TCP to serial/serial to TCP server
- ColdFire\_Lite\_TCP\_with\_Web\_Server  
Web (HTTP) server with dynamic HTML

### 5.3 ColdFire\_Lite\_TCP\_alarm

The alarm demo application includes both PC-side and ColdFire-side firmware. This code is an example of a remote sensor application, where a remote sensor periodically sends data over TCP to a host server.



### 5.4 HTTP Client Firmware

The HTTP client provides the ability to read web pages and XML data from the Internet using a ColdFire processor. The HTTP client uses the DHCP client to automatically acquire an IP address and other TCP/IP information, including the IP address of any DNS server. Then the HTTP client uses the DNS client to translate any user-provided URLs into IP addresses.

The HTTP client uses the GET method to request a page from the server. Along with the GET request is the HTTP header. The HTTP header is hard-coded in the HTTP client via constant strings declared in the file `emg_http_client.c`.

## 5.5 Wget Command – An Example of Using the HTTP Client

The Wget command is a command often found in Linux distributions that transfer files using the HTTP protocol. The Wget command is a console-based HTTP client. Using the menuing system provided by the ColdFire TCP/IP stack (explained in application note AN3470) and the HTTP client, Wget functionality can be added to the ColdFire TCP/IP stack.

## 5.6 RSS/XML Character Data Filter

To extract the character data (the information you actually want to read) from the RSS stream, all the meta-text must be filtered out. Any valid HTML must be translated and processed. For instance, the HTML tag that causes a line break is `<br>`. This would appear as `&lt;br&gt;` in the RSS stream. The filter must correctly translate `&lt;br&gt;` into a carriage return and line feed.

Other HTML tags that are routinely embedded in character data include paragraph tags `<p>` and image tags `<img>`. In the stream these tags appear as `&lt;p&gt;` and `&lt;img&gt;` respectively. The paragraph tag can be translated to a carriage return and line feed, but the image tag must be ignored unless the embedded system can process images.

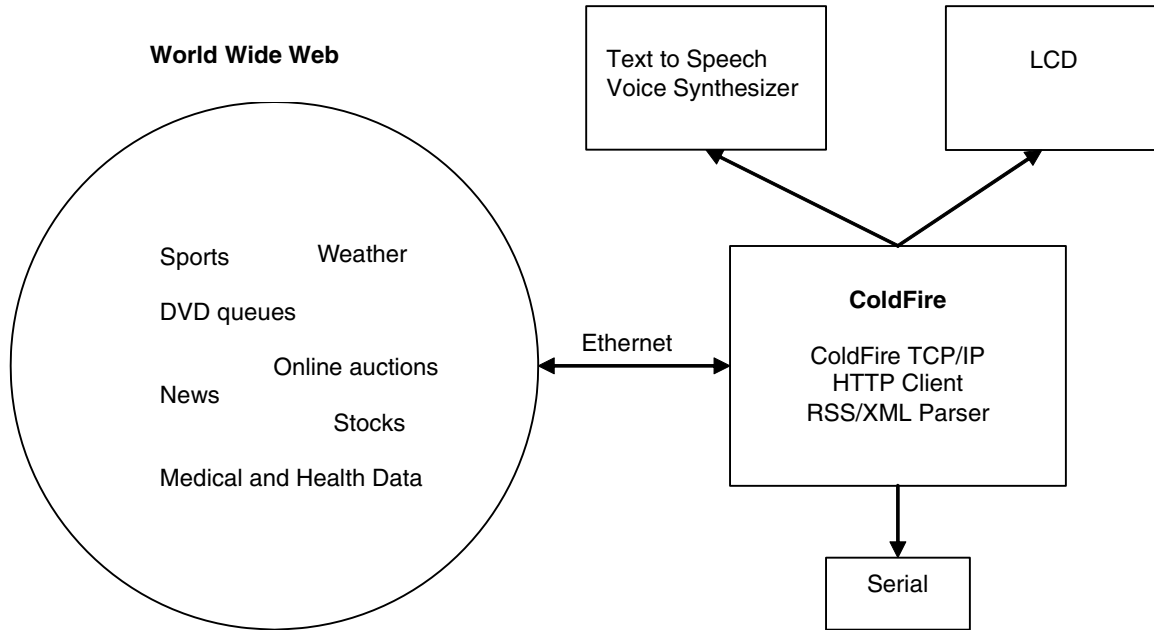
The filter takes in an XML or RSS data stream and a list of tags. It outputs only the character data from the selected tags. The tag list is an array of pointers to those tag strings that need to be filtered.

Normally the filter returns 0. When the filter processes the “>” in a tag in the list, it returns to the filter array the index + 1 for the tag that it found. Example: after detecting a `<title>` tag in an RSS or XML stream the filter will return 1. After detecting a `<description>` tag in a stream the filter will return 2. Normally the filter returns 0.

# 6 Example Embedded Appliance — RSS/XML Feed Reader

The RSS/XML feed reader is an embedded appliance that allows users to display and even hear real-time content from the World Wide Web. The purpose of the embedded appliance is to provide instant real-time information without booting a PC. There are many types of real-time data available on the web; weather data (current and forecast), online DVD queue data, online auction data, sports score data, real-time news data, real-time stock data, medical and health data, and much more. All this data is available on the web as either an XML feed or an RSS feed. This appliance connects to the web, gets the desired feed, and parses the text information or character data from the feed. That data is displayed on the LCD, sent to the serial port, and spoken through the speech processor.

For complete details on the RSS feed reader please see AN3518, *Advanced ColdFire TCP/IP Clients*.



## 6.1 M52233DEMO Board from Freescale Semiconductor

The M52233DEMO board is a reference board used to evaluate Freescale's ColdFire MC52233 processor. The inexpensive board includes a serial port, USB BDM debug port, and Ethernet port. The board along with the free CodeWarrior tools (up to 128K of flash) are all you need to get up and running on your Ethernet projects. Freescale provides a free public source TCP/IP stack on their website. This TCP/IP stack

is what the application in this article runs on. The ColdFire TCP/IP stack is documented thoroughly in application note AN3470.

The demo board includes a 40-pin header connector, giving the user access to most of the signals from the ColdFire microcontroller. The demo board includes a 3-axis accelerometer connected to three of the ColdFire analog ports. The board also includes a potentiometer and two user buttons.

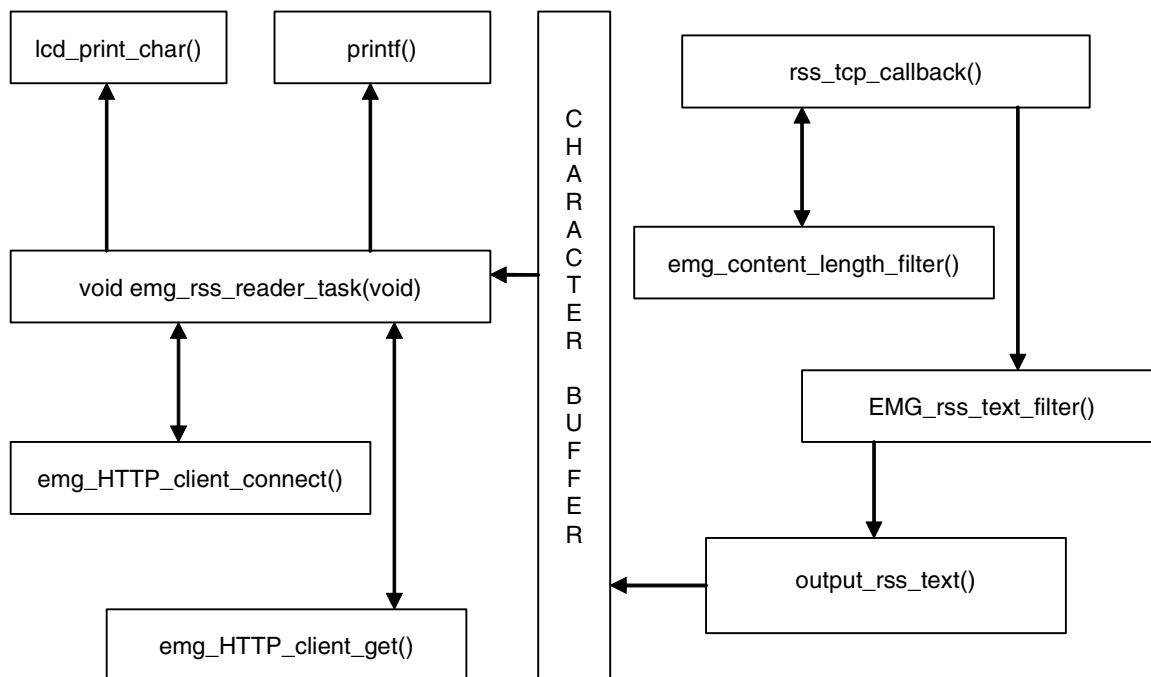
## 6.2 LCD

The parallel LCD is a 4 × 20 character display that uses the standard Hitachi instruction set. The LCD is used in its 4-bit mode, requiring only six connections to the micro, the 4-bit data bus, a clock signal (E), and a register select line (RS). The firmware includes a library to drive the LCD.

## 6.3 Voice Synthesizer

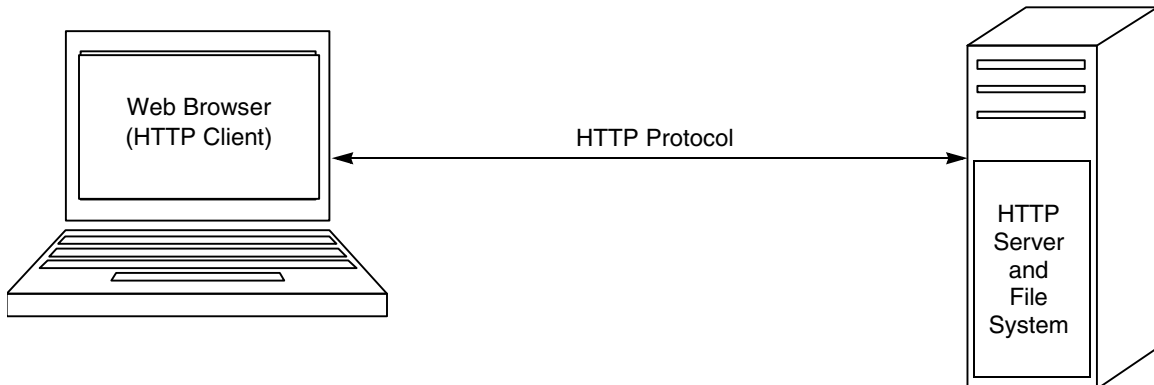
The RC Systems V-Stamp voice synthesizer is an easy-to-use text-to-speech processor. The V-Stamp is a fully self-contained module, requiring only power, a speaker, a resistor, two capacitors, and a serial connection to an embedded system. The V-Stamp communicates with the embedded system using a UART. The module automatically sets its baud rate to that of the embedded system. From both a hardware and firmware point of view, there is very little work required to add the V-Stamp module to the RSS feed reader.

## 6.4 Firmware



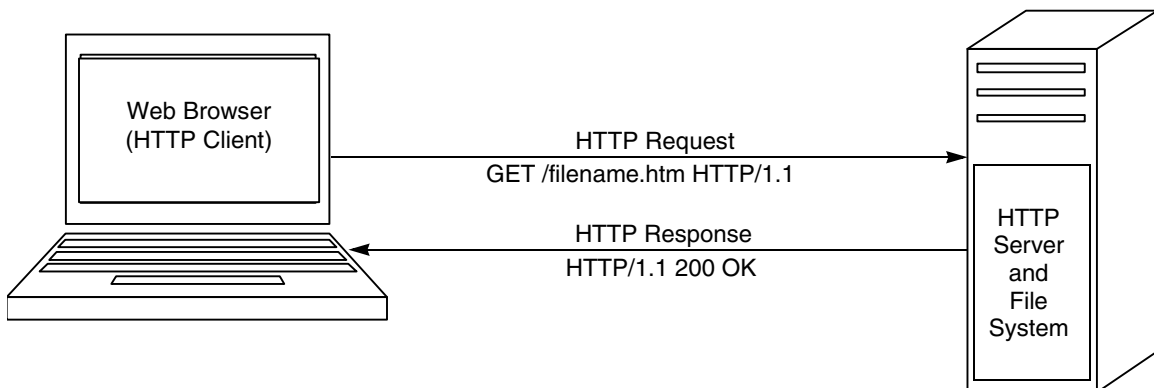
## 6.5 HTTP Protocol

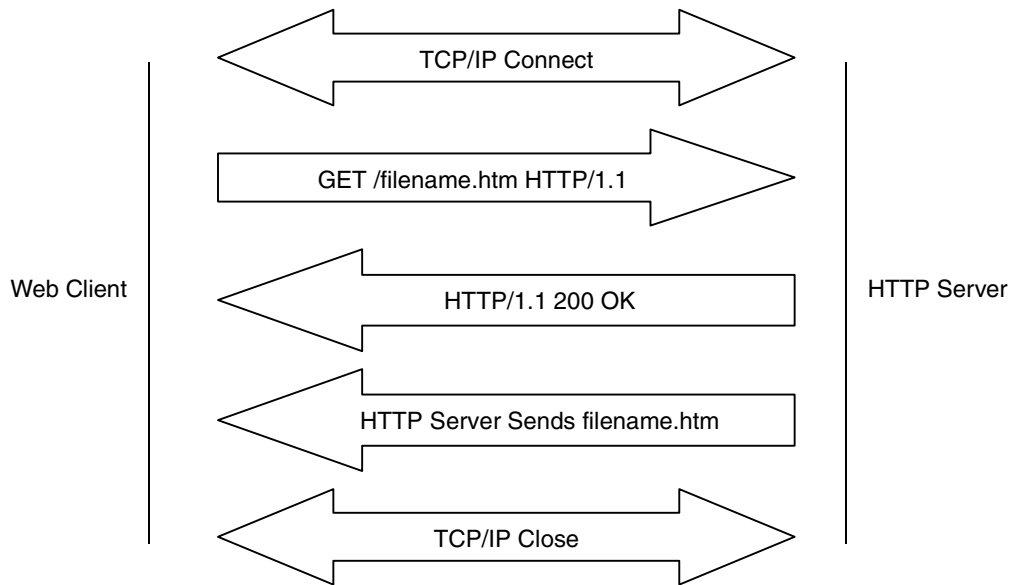
HTTP (Hypertext Transfer Protocol) is the communication protocol of the World Wide Web. HTTP is used to transfer web pages (hypertext documents) across the Internet. An HTTP connection has two parts, the HTTP client (web browser) and the HTTP server. The HTTP client is used to receive and view the web pages. The HTTP server is used to store, organize, and transfer the web pages.



HTTP is defined by RFC 1945 and RFC 2616. RFC 1945 defines HTTP 1.0, and RFC 2616 defines the latest version, HTTP 1.1.

HTTP is a request-response protocol. The client requests a web page from the server and the server responds with the web page contents. HTTP can be used to send any type of data, including binary data. The client requests a file using the GET method (HTTP is an ASCII protocol). The server responds with an HTTP header followed by the file contents. Within the request, the version number of the HTTP is also embedded in ASCII. This tells the server the limitations of the client.





## 6.6 Really Simple Syndication (RSS)

RSS feeds are available everywhere on the Internet. The idea behind the RSS feed is to convey dynamic textual information in a simple standard format.

RSS originated in 1999 with the idea of providing content in a simple easy-to-understand format. Instead of describing a complete document in the way that HTML does, RSS feeds use XML to describe data. An RSS feed is simply an XML document containing data. The methods used to convey the data within the XML document are described in the RSS 2.0 specification. All RSS files must conform to the XML 1.0 specification. RSS feeds generally use HTTP as the transport mechanism.

## 6.7 Extensible Markup Language (XML)

The XML 1.0 specification can be found at [www.w3.org/TR/REC-xml/](http://www.w3.org/TR/REC-xml/).

XML is a language used to describe and parse information. It is very similar to structures in C.

Data is organized into elements, with each element assigned to a tag. The data in the element is surrounded by a start tag and an end tag. The name in the start and end tags defines the element's type. The end tag name must be the same as the start tag name, except the end tag is identified by the addition of a "/" before the tag name.

### 6.7.1 Tags

Here is an example of an XML tag:

```
<TITLE>Advanced ColdFire TCP/IP Clients</TITLE>
```

TITLE is the type, <TITLE> is the start tag, and </TITLE> is the end tag. The data is between the tags. Just like a C data structure, the data is associated with the type. The data between the start and end tags is referred to as the element's content.

Elements can contain other elements, which provides a method of grouping data of different types under a single name. Just like a C structure, the particular piece of data is referenced by specifying a path to the data.

```
<APPNOTES>
  <BYEG>
    <AN3455>ColdFire HTTP Server</AN3455>
    <AN3470>ColdFire TCP/IP Stack</AN3470>
    <AN3492>ColdFire USB Stack</AN3492>
  </BYEG>
</APPNOTES>
```

## 6.7.2 Special Characters and Escape Sequences

The “&”, “<”, and “>” characters are special XML characters. They are used as indicated above, to define XML tags and escape sequences. Escape sequences are a method of specifying a character using a code, as opposed to using a single symbol. Escape sequences start with an ampersand (&) and end with a semicolon(;).

Because XML ordinarily uses these characters for special functions, here is how to have them appear in XML data:

- “<” is indicated using the escape sequence &lt;
- “>” is indicated using the escape sequence &gt;
- “&” is indicated using the escape sequence &amp;

## 6.7.3 CDATA Sections: The Exception to the Rule

All rules have exceptions, and in XML it's the CDATA section. Any text specified in a CDATA section is ignored by the XML parser, allowing the use of special characters without the need for escape sequences. A CDATA section starts with a “<!CDATA[” string, and is terminated with a “]]>” string. Anything between the brackets is ignored by the XML parser.

- <!CDATA[ character data here is ignored by the XML parser ]]>
- <!CDATA[ <THIS\_IS\_NOT\_A\_TAG> &lt; ]]>
- Because the text between the brackets is ignored by the XML parser, both the tag and the escape sequence are not parsed, but instead are interpreted as text or character data.

## 6.7.4 Finding Text or Character Data in an XML Document

From the XML 1.0 specification: “All text that is not markup constitutes the character data of the document.” That would include all the text between the brackets in a CDATA section, and any text not

between “<” and “>” brackets in the main body. Escape sequences in the main body represent a single piece of character data.

To filter out the character data from an XML document, simply remove all tags as well as “<” and “>” brackets. Then translate the escape sequences into actual characters.

### 6.7.5 Sample XML File

In the sample XML document below, notice how data is encapsulated by tags, and how the tag names describe the data. It even looks like a C structure. To find the desired data in an XML document, simply look for the start and end tags with the name of the type of data you are looking for. The actual data associated with the desired type is between the tags.

[http://www.weather.gov/data/current\\_obs/KUGN.xml](http://www.weather.gov/data/current_obs/KUGN.xml)

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<current_observation version="1.0" xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="http://www.weather.gov/data/current_obs/current_observation.xsd">
  <credit>NOAA's National Weather Service</credit>
  <credit_URL>http://weather.gov/</credit_URL>
  <image>
    <url>http://weather.gov/images/xml_logo.gif</url>
    <title>NOAA's National Weather Service</title>
    <link>http://weather.gov</link>
  </image>
  <suggested_pickup>15 minutes after the hour</suggested_pickup>
  <suggested_pickup_period>60</suggested_pickup_period>
  <location>Chicago / Waukegan, Waukegan Regional Airport, IL</location>
  <station_id>KUGN</station_id>
  <latitude>42.420</latitude>
  <longitude>-87.870</longitude>
  <observation_time>Last Updated on Jul 28, 10:52 am CDT</observation_time>
  <observation_time_rfc822>Sat, 28 Jul 2007 10:52:00 -0500 CDT</observation_time_rfc822>
  <weather>Overcast</weather>
  <temperature_string>73 F (23 C)</temperature_string>
  <temp_f>73</temp_f>
  <temp_c>23</temp_c>
  <relative_humidity>81</relative_humidity>
  <wind_string>From the Northeast at 13 Gusting to 21 MPH</wind_string>
  <wind_dir>Northeast</wind_dir>
  <wind_degrees>30</wind_degrees>
  <wind_mph>12.65</wind_mph>
```

```

<wind_gust_mph>21</wind_gust_mph>
<pressure_string>29.98" (1014.1 mb)</pressure_string>
<pressure_mb>1014.1</pressure_mb>
<pressure_in>29.98</pressure_in>
<dewpoint_string>67 F (19 C)</dewpoint_string>
<dewpoint_f>67</dewpoint_f>
<dewpoint_c>19</dewpoint_c>
<heat_index_string>73 F (23 C)</heat_index_string>
<heat_index_f>73</heat_index_f>
<heat_index_c>23</heat_index_c>
<windchill_string>NA</windchill_string>
<windchill_f>NA</windchill_f>
<windchill_c>NA</windchill_c>
<visibility_mi>10.00</visibility_mi>
<icon_url_base>http://weather.gov/weather/images/fcicons/</icon_url_base>
<icon_url_name>ovc.jpg</icon_url_name>

<two_day_history_url>http://www.weather.gov/data/obhistory/KUGN.html</two_day_his
tory_url>
  <ob_url>http://www.nws.noaa.gov/data/METAR/KUGN.1.txt</ob_url>
  <disclaimer_url>http://weather.gov/disclaimer.html</disclaimer_url>
  <copyright_url>http://weather.gov/disclaimer.html</copyright_url>
  <privacy_policy_url>http://weather.gov/notice.html</privacy_policy_url>
</current_observation>

```

## 6.7.6 Drawback of XML Documents

The problem with XML documents is the flexibility of the tag names. The creator of the XML document can use any name to describe the data. To avoid confusion, a standard is required to standardize the tag names, and the way that data is associated with a specific type. That standard exists and is called the RSS specification.

## 6.8 The RSS Specification

RSS is a dialect of XML — it embeds HTML constructs into the XML architecture. RSS also defines a set group of elements and a general template using those elements. There are many elements defined in the specification, but here we will concentrate on the elements of interest for the RSS appliance.

### 6.8.1 Typical RSS File

```

<channel>
  <title>The name of the channel</title>
  <link>URL to the HTML website corresponding to this channel</link>

```

```

<description>Text describing the channel</description>
<item>
  <title>Item1 Title</title>
  <link>URL of item 1</link>
  <description>Text for item 1</description>
</item>
<item>
  <title>Item2 Title</title>
  <link>URL of item 2</link>
  <description>Text for item 2</description>
</item>
</channel>

```

The organization of the RSS file can be compared to a newspaper. The channel is the name of the paper, the items are the articles in the paper, the titles are the titles of the articles, and the descriptions are the text of the articles.

## 6.8.2 Sample RSS File

The sample RSS file given here contains the same data as the sample XML file above. Notice the structure and tag names. The RSS standard defines the title and description tag names, and those tags contain the data we want. All RSS 2.0 compliant feeds will contain title and description tags.


[http://www.weather.gov/data/current\\_obs/KUGN.rss](http://www.weather.gov/data/current_obs/KUGN.rss)

```

<?xml version="1.0" encoding="ISO-8859-1" ?>
= <rss version="2.0" xmlns:dc="http://purl.org/dc/elements/1.1/">
= <channel>
  <title>Weather at Chicago / Waukegan, Waukegan Regional Airport, IL - via NOAA's
National Weather Service</title>
  <link>http://www.weather.gov/data/current_obs/</link>
  <lastBuildDate>Sat, 28 Jul 2007 16:32:11 UT</lastBuildDate>
  <ttl>60</ttl>
  <description>Weather conditions from NOAA's National Weather Service.</description>
  <language>en-us</language>
  <managingEditor>robert.bunge@noaa.gov</managingEditor>
  <webMaster>w-nws.webmaster@noaa.gov</webMaster>
= <image>
  <url>http://www.weather.gov/images/xml_logo.gif</url>
  <title>NOAA - National Weather Service</title>
  <link>http://www.weather.gov/data/current_obs/</link>
  </image>
= <item>

```

```

<title>Overcast and 73 degrees F at Chicago / Waukegan, Waukegan Regional Airport,
IL</title>
<link>http://weather.noaa.gov/weather/current/KUGN.html</link>
<description>
- <![CDATA[
<br />
]]> 
Winds are Northeast at 13 Gusting to 21 MPH. The pressure is 29.98" (1014.1 mb) and
the humidity is 81%. The heat index is 73. Last Updated on Jul 28, 10:52 am CDT.
</description>
<guid isPermaLink="false">Sat, 28 Jul 2007 10:52:00 -0500 CDT</guid>
</item>
</channel>
</rss>

```

## 6.9 RSS/XML Character Data Filter

To extract the character data (the information you actually want to read) from the RSS stream, all the meta-text must be filtered out. See [Section 5.6, “RSS/XML Character Data Filter,”](#) for more information.

### 6.9.1 RSS/XML Character Data Filter State Machine

The character data filter is implemented as a finite state machine. The state machine uses only two global variables, a state variable and a FILO. The purpose of the FILO is to store characters. Each byte entered into the filter is shifted left into the FILO. Therefore the most recent character is at location `filo_buff[FILO_BUFF_SIZE-1]`. The FILO buffer size must be larger than the largest tag name expected. The FILO buffer size is set with the `FILO_BUFF_SIZE` macro.

```
#define FILO_BUFF_SIZE 32
```

States:

```

STATE_ZERO
STATE_TAGSEARCH
STATE_INTAG
STATE_PRINT
STATE_SKIP
STATE_SKIP_NON_ASCII
STATE_SKIP_SPACE
STATE_CDATA
STATE_CDATA_PRINT
STATE_CDATA_SKIP_AMP

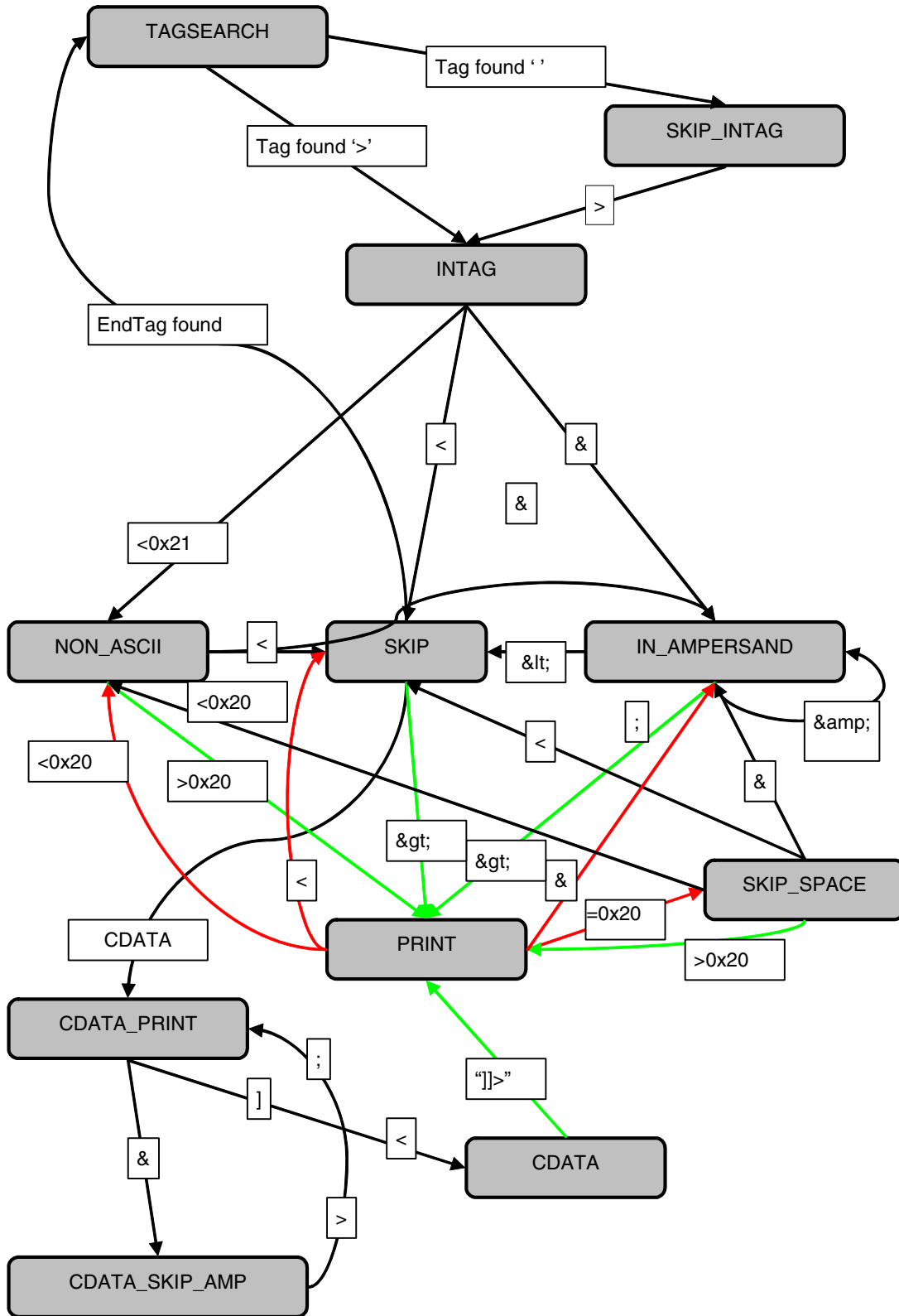
```

```
STATE_SKIP_INTAG  
STATE_IN_AMPERSAND
```

Global Variables:

```
unsigned char filo_buff[FILO_BUFF_SIZE];  
unsigned char state;
```

```
unsigned char EMG_rss_text_filter(unsigned char data, unsigned char **tag_filter)
```



The filter takes in an XML or RSS data stream and a list of tags. It outputs only the character data from the selected tags. The tag list is an array of pointers to those tag strings that need to be filtered.

Normally the filter returns 0. When the filter processes the “>” in a tag in the list, it returns into the filter array the index+1 for the tag that it found. Example: after detecting a <title> tag in an RSS or XML stream the filter will return 1. After detecting a <description> tag in a stream the filter will return 2. The filter returns 0 if a tag is not found.

```
Const unsigned char *tag_filter[] =
{
    {(const unsigned char *)"title"},
    {(const unsigned char *)"description"},
    {(const unsigned char *)""}
};
```

Figure 7. Sample tag\_filter Pointer Array

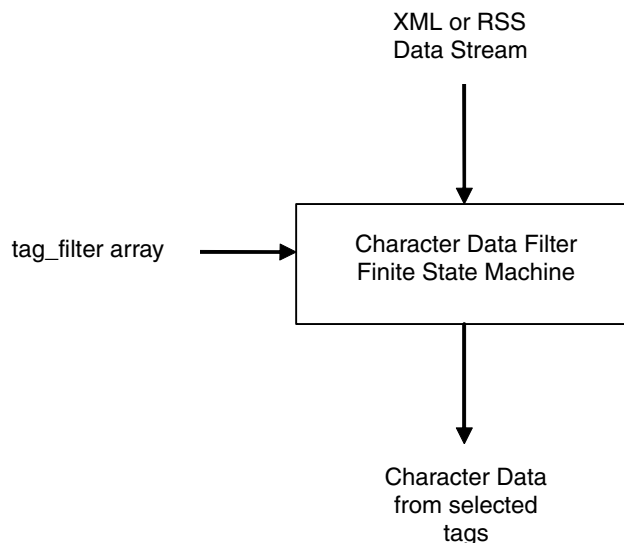


Figure 8. Character Data Filter Usage

## 6.10 Using the RSS/XML Feed Reader

Using the RSS/XML feed reader firmware is easy:

1. Set the variable url to the URL or to the desired RSS or XML server.

```
static const unsigned char url[] = "http://www.weather.gov/data/current_obs/KUGN.rss";
```

2. Set the tag\_filter[] to the type of data you want to display.

For RSS feeds, <title> and <description> would be a first choice.

For XML feeds, this could be any tag name depending on the information.

## Conclusion

```
const unsigned char *tag_filter[] =
{
    {(const unsigned char *)"title"},
    {(const unsigned char *)"description"},
    {(const unsigned char *)""}
};
```

3. Set the character buffer size large enough to collect your filtered data.

```
#define    RSS_CHARACTER_BUFF_SIZE2048
```

4. Compile the project and flash it to the board.

After the TCP/IP stack comes up, these actions occur:

1. The DHCP client automatically acquires an IP address and DNS IP address.
2. It displays and speaks a title screen.
3. It connects to the server specified in the URL.
4. The status of the connection is displayed and spoken.
5. The file is downloaded from the server using the HTTP client.
6. After the connection closes, the file is displayed and spoken.
7. The RSS/XML feed reader sleeps, waiting for either SW1 or SW2 to be pushed. When that happens, a connection is initiated to the server specified in the URL that starts the download/display/speak process all over again.

## 6.11 XML Streams

For XML streams, the `EMG_rss_text_filter()` return value is used to determine which tag is applied to the data that comes after the tag. This allows another const array containing more descriptive names to be used to describe the data from the tag. Simply use the `EMG_rss_text_filter()` return value-1 to index into a descriptive name array, then send the indexed string into the character buffer before leaving the RSS callback function. The XML filter will then place the data from the tag in the character filter directly after the descriptive name.

## 7 Conclusion

Adding Ethernet to embedded products provides a level of connectivity never before available in the embedded space. Embedded devices can be networked together using the same cables and hardware used by the industry to connect PCs together. The embedded device can become part of the PC network. Ethernet and TCP/IP enable the embedded device to connect to the ultimate network, the Internet. After the embedded device is connected to the Internet it is available to the world, and controllable or viewable from anywhere. The possibilities are endless, from remote monitoring and control to distributed control, and even real-time world-wide data presentation through the RSS feed reader.



THIS PAGE IS INTENTIONALLY BLANK

## How to Reach Us:

### Home Page:

[www.freescale.com](http://www.freescale.com)

### Web Support:

<http://www.freescale.com/support>

### USA/Europe or Locations Not Listed:

Freescale Semiconductor, Inc.  
Technical Information Center, EL516  
2100 East Elliot Road  
Tempe, Arizona 85284  
+1-800-521-6274 or +1-480-768-2130  
[www.freescale.com/support](http://www.freescale.com/support)

### Europe, Middle East, and Africa:

Freescale Halbleiter Deutschland GmbH  
Technical Information Center  
Schatzbogen 7  
81829 Muenchen, Germany  
+44 1296 380 456 (English)  
+46 8 52200080 (English)  
+49 89 92103 559 (German)  
+33 1 69 35 48 48 (French)  
[www.freescale.com/support](http://www.freescale.com/support)

### Japan:

Freescale Semiconductor Japan Ltd.  
Headquarters  
ARCO Tower 15F  
1-8-1, Shimo-Meguro, Meguro-ku,  
Tokyo 153-0064  
Japan  
0120 191014 or +81 3 5437 9125  
[support.japan@freescale.com](mailto:support.japan@freescale.com)

### Asia/Pacific:

Freescale Semiconductor Hong Kong Ltd.  
Technical Information Center  
2 Dai King Street  
Tai Po Industrial Estate  
Tai Po, N.T., Hong Kong  
+800 2666 8080  
[support.asia@freescale.com](mailto:support.asia@freescale.com)

### For Literature Requests Only:

Freescale Semiconductor Literature Distribution Center  
P.O. Box 5405  
Denver, Colorado 80217  
1-800-441-2447 or 303-675-2140  
Fax: 303-675-2150  
[LDCForFreescaleSemiconductor@hibbertgroup.com](mailto:LDCForFreescaleSemiconductor@hibbertgroup.com)

Document Number: COLDEtherWP  
Rev. 0  
07/2008

Information in this document is provided solely to enable system and software implementers to use Freescale Semiconductor products. There are no express or implied copyright licenses granted hereunder to design or fabricate any integrated circuits or integrated circuits based on the information in this document.

Freescale Semiconductor reserves the right to make changes without further notice to any products herein. Freescale Semiconductor makes no warranty, representation or guarantee regarding the suitability of its products for any particular purpose, nor does Freescale Semiconductor assume any liability arising out of the application or use of any product or circuit, and specifically disclaims any and all liability, including without limitation consequential or incidental damages. "Typical" parameters that may be provided in Freescale Semiconductor data sheets and/or specifications can and do vary in different applications and actual performance may vary over time. All operating parameters, including "Typicals", must be validated for each customer application by customer's technical experts. Freescale Semiconductor does not convey any license under its patent rights nor the rights of others. Freescale Semiconductor products are not designed, intended, or authorized for use as components in systems intended for surgical implant into the body, or other applications intended to support or sustain life, or for any other application in which the failure of the Freescale Semiconductor product could create a situation where personal injury or death may occur. Should Buyer purchase or use Freescale Semiconductor products for any such unintended or unauthorized application, Buyer shall indemnify and hold Freescale Semiconductor and its officers, employees, subsidiaries, affiliates, and distributors harmless against all claims, costs, damages, and expenses, and reasonable attorney fees arising out of, directly or indirectly, any claim of personal injury or death associated with such unintended or unauthorized use, even if such claim alleges that Freescale Semiconductor was negligent regarding the design or manufacture of the part.

RoHS-compliant and/or Pb-free versions of Freescale products have the functionality and electrical characteristics as their non-RoHS-compliant and/or non-Pb-free counterparts. For further information, see <http://www.freescale.com> or contact your Freescale sales representative.

For information on Freescale's Environmental Products program, go to <http://www.freescale.com/epp>.

Freescale™ and the Freescale logo are trademarks of Freescale Semiconductor, Inc. All other product or service names are the property of their respective owners.  
© Freescale Semiconductor, Inc. 2008. All rights reserved.

