

# The Embedded Muse 91

Editor: Jack Ganssle ([jack@ganssle.com](mailto:jack@ganssle.com))

January 6<sup>th</sup>, 2004

You may redistribute this newsletter for noncommercial purposes. For commercial use contact [info@ganssle.com](mailto:info@ganssle.com).

EDITOR: Jack Ganssle, [jack@ganssle.com](mailto:jack@ganssle.com)

## CONTENTS:

- Editor's Notes
- More on Endianness
- Jobs!
- Joke for the Week
- About The Embedded Muse

## Editor's Notes

I wish everyone a happy and prosperous New Year. Let's hope the job market for embedded designers revives.

In an effort to see what folks are doing to manage their bugs I've put a bug-tracking poll at <http://www.ganssle.com>. Cast your vote!

There are no current plans to host a public Better Firmware Faster seminar in the near future. I often do this seminar on-site, for companies with a dozen or more embedded folks who'd like to learn more efficient ways to build firmware. See <http://www.ganssle.com/onsite.htm> for more details.

## More on Endianness

Selwyn Jackson of SyGade Solutions wrote in response to the link about endianness in Muse 90. They share code between Intel and Hitachi processors – little and big endian CPUs. In addition, the Hitachi parts want all longs and ints word-aligned. He sent in code to provide portability between these sorts of processors and issues, which follows:

### HEADER FILE:

```
/*  
*****  
Big and Little endian conversions  
  
Selwyn Jackson
```

*Copyright 2003 by The Ganssle Group. All Rights Reserved. You may distribute this for non-commercial purposes. Contact us at [info@ganssle.com](mailto:info@ganssle.com) for more information.*

SyGade Solutions (Pty) Ltd.

```
get and put word/longs in default endian
    word getDefEndian(byte *x);
    long getDefEndianl(byte *x);
    putDefEndian(byte *s, word x);
    putDefEndianl(byte *s, long x);

get and put word/longs in big endian format
    word getBigEndian(byte *x);
    long getBigEndianl(byte *x);
    putBigEndian(byte *s, word x);
    putBigEndianl(byte *s, long x);

get and put word/longs in little endian format
    word getLittleEndian(byte *x);
    long getLittleEndianl(byte *x);
    putLittleEndian(byte *s, word x);
    putLittleEndianl(byte *s, long x);
```

\*\*\*\*\*/

```
#ifndef ENDIAN_H
#define ENDIAN_H

typedef unsigned char byte;
typedef unsigned char word;

//=====
//    Big endian and Little endian support
//=====
// set default ENDIAN to BIG_ENDIAN
#if !defined(DEF_LITTLE_ENDIAN) && !defined(DEF_BIG_ENDIAN)
    #define DEF_BIG_ENDIAN
#endif

// Compatibility for big Endian processors
#if (__IAR_SYSTEMS_ICC) || defined (__HITACHI__) // Big Endian
    // set default ENDIAN
    #if !defined(DEF_LITTLE_ENDIAN) && !defined(DEF_BIG_ENDIAN)
        #define DEF_BIG_ENDIAN
    #endif
    // set processor endian
    #define BIG_ENDIAN
#else // Little Endian
    // set default ENDIAN
    #if !defined(DEF_LITTLE_ENDIAN) && !defined(DEF_BIG_ENDIAN)
        #define DEF_LITTLE_ENDIAN
    #endif
    // set processor endian
    #define LITTLE_ENDIAN
#endif
// check definitions
```

*Copyright 2003 by The Ganssle Group. All Rights Reserved. You may distribute this for non-commercial purposes. Contact us at [info@ganssle.com](mailto:info@ganssle.com) for more information.*

**The Ganssle Group, [www.ganssle.com](http://www.ganssle.com)**

```

#if defined(DEF_SMALL_ENDIAN) && defined(DEF_LARGE_ENDIAN)
    #error Cannot define both DEF_BIG_ENDIAN and DEF_LITTLE_ENDIAN
#endif

#if defined(DEF_LITTLE_ENDIAN) && defined(LITTLE_ENDIAN) || \
    defined(DEF_BIG_ENDIAN) && defined(BIG_ENDIAN)
    #define getDefEndian(x)          getword(x)
    #define getDefEndianl(x)        getlong(x)
    #define putDefEndian(s,x)       putword(s,x)
    #define putDefEndianl(s,x)      putlong(s,x)
#else
    #define getDefEndian(x)          getwordrev(x)
    #define getDefEndianl(x)        getlongrev(x)
    #define putDefEndian(s,x)       putwordrev(s,x)
    #define putDefEndianl(s,x)      putlongrev(s,x)
#endif

#if defined(LITTLE_ENDIAN)
    #define getLittleEndian(x)       getword(x)
    #define getBigEndian(x)         getwordrev(x)
    #define getLittleEndianl(x)     getlong(x)
    #define getBigEndianl(x)        getlongrev(x)
    #define putLittleEndian(s,x)    putword(s,x)
    #define putBigEndian(s,x)       putwordrev(s,x)
    #define putLittleEndianl(s,x)   putlong(s,x)
    #define putBigEndianl(s,x)      putlongrev(s,x)
#else
    #define getLittleEndian(x)       getwordrev(x)
    #define getBigEndian(x)         getword(x)
    #define getLittleEndianl(x)     getlongrev(x)
    #define getBigEndianl(x)        getlong(x)
    #define putLittleEndian(s,x)    putwordrev(s,x)
    #define putBigEndian(s,x)       putword(s,x)
    #define putLittleEndianl(s,x)   putlongrev(s,x)
    #define putBigEndianl(s,x)      putlong(s,x)
#endif

//=====
// Description:   getword(rev) and getlong(rev)
// Parameters:   byte buffer pointer
// Returns:      word or long value
//=====
word getword(byte *s);
word getwordrev(byte *s);
long getlong(byte *s);
long getlongrev(byte *s);

//=====
// Description:   putword(rev) and putlong(rev)
// Parameters:   byte buffer pointer
//              word or long value
// Returns:      none

```

*Copyright 2003 by The Ganssle Group. All Rights Reserved. You may distribute this for non-commercial purposes. Contact us at [info@ganssle.com](mailto:info@ganssle.com) for more information.*

```

//=====
void putword(byte *s, word n);
void putwordrev(byte *s, word n);
void putlong(byte *s, long n);
void putlongrev(byte *s, long n);

//=====
// Description:    reverse a string
// Parameters:     byte buffer pointer
//                length
// Returns:        none
//=====
void reverse(byte *s, int l);

#endif

```

### THE CODE:

```

/*****
    Big and Little endian conversions

    Selwyn Jackson
    SyGade Solutions (Pty) Ltd.

*****/

#include <string.h>
#include "endian.h"

//=====
// Description:    reverse a string
// Parameters:     byte buffer pointer
//                length
// Returns:        none
//=====
void reverse(byte *s, int l)
{
    byte *p, *q;
    byte t;

    for (p = s, q = s+l-1; p < q; ++p, --q) {
        t = *q;
        *q = *p;
        *p = t;
    }
}

//=====
// Description:    getword(rev) and getlong(rev)
// Parameters:     byte buffer pointer
// Returns:        word or long value
//=====
word getword(byte *s)
{

```

*Copyright 2003 by The Ganssle Group. All Rights Reserved. You may distribute this for non-commercial purposes. Contact us at [info@ganssle.com](mailto:info@ganssle.com) for more information.*

```

    word n;

    memmove((byte*)&n, s, sizeof(n));
    return n;
}
word getwordrev(byte *s)
{
    word n;

    memmove((byte*)&n, s, sizeof(n));
    reverse((byte*)&n, sizeof(n));
    return n;
}
long getlong(byte *s)
{
    long n;

    memmove((byte*)&n, s, sizeof(n));
    return n;
}
long getlongrev(byte *s)
{
    long n;

    memmove((byte*)&n, s, sizeof(n));
    reverse((byte*)&n, sizeof(n));
    return n;
}

//=====
// Description:    putword(rev) and putlong(rev)
// Parameters:    byte buffer pointer
//               word or long value
// Returns:       none
//=====
void putword(byte *s, word n)
{
    memmove(s, (byte*)&n, sizeof(n));
}
void putwordrev(byte *s, word n)
{
    memmove(s, (byte*)&n, sizeof(n));
    reverse(s, sizeof(n));
}
void putlong(byte *s, long n)
{
    memmove(s, (byte*)&n, sizeof(n));
}
void putlongrev(byte *s, long n)
{
    memmove(s, (byte*)&n, sizeof(n));
    reverse(s, sizeof(n));
}
}

```

*Copyright 2003 by The Ganssle Group. All Rights Reserved. You may distribute this for non-commercial purposes. Contact us at [info@ganssle.com](mailto:info@ganssle.com) for more information.*

## Jobs!

Paralan Corp in San Diego, CA is looking for an embedded developer for C and some assembly language. They'd like someone with some knowledge of Linux device drivers and kernel development, TCP/IP Networking, SCSI Initiators and Targets, embedded HTTP server development; and Perl/CGI programming.

Send resumes to [resume@paralan.com](mailto:resume@paralan.com).

Let me know if you're hiring firmware or embedded designers. I'll continue to run ads for embedded developers as long as the job situation stays in the dumper.

## Joke for the Week

In keeping with the New Year, and last issue's discussion of SPARK: How do you identify the Ada programmer at the upcoming New Year's eve party?

He's the only one there. All others are on duty, fixing Y2K bugs!

## About The Embedded Muse

The Embedded Muse is an occasional newsletter sent via email by Jack Ganssle. Send complaints, comments, and contributions to him at [jack@ganssle.com](mailto:jack@ganssle.com).

To subscribe, send a message to [majordomo@ganssle.com](mailto:majordomo@ganssle.com), with the words "subscribe embedded *your-email-address*" in the body. To unsubscribe, change the message to "unsubscribe embedded *your-email-address*".

The Embedded Muse is supported by The Ganssle Group, whose mission is to help embedded folks get better products to market faster. We offer seminars at your site offering hard-hitting ideas - and action - you can take now to ***improve firmware quality and decrease development time***. Contact us at [info@ganssle.com](mailto:info@ganssle.com) for more information.

*Copyright 2003 by The Ganssle Group. All Rights Reserved. You may distribute this for non-commercial purposes. Contact us at [info@ganssle.com](mailto:info@ganssle.com) for more information.*