PBXProj Identifiers

If you have ever looked inside of a `pbxproj` file you will know it looks something like this:

```
// !$*UTF8*$!
{
    archiveVersion = 1;
    classes = {
    };
    objectVersion = 46;
    objects = {

/* Begin PBXBuildFile section */
        1548E7061BE65D16001A0E2F /* AppDelegate.m in Sources */ = {isa
        1548E7091BE65D16001A0E2F /* main.m in Sources */ = {isa = PBXBu
        1548E70B1BE65D16001A0E2F /* Assets.xcassets in Resources */ = {
...
/* End XCConfigurationList section */
    };
    rootObject = 1548E6F91BE65D16001A0E2F /* Project object */;
}
```

Each object in the pbxproj file has a unique identifier. These are created by Xcode in such a way that won't cause conflicts. Xcode builds a unique identifier based on a couple of pieces of metadata on creation of the object it is associated with.

```
struct globalidentifier {
    int8_t user;            // encoded username
    int8_t pid;             // encoded current pid #
    int16_t _random;        // encoded random value
    int32_t _time;          // encoded time value
    int8_t zero;            // zero
    int8_t host_shift;      // encoded, shifted hostid
    int8_t host_h;          // high byte of lower bytes of hostid
    int8_t host_l;          // low byte of lower bytes of hostid
} __attribute__((packed));  // packed, so we always have a length of 12
```

This information created once, then stored as a global variable to prevent overhead of the same information being encoded again. It get resused and the random and time values will get updated each time an identifier must be created.

```
// globals
uint64_t packedValueForChar = 0x1f1f1f1f1f1f1f1f;
int8_t hasInitialized = false;
int32_t lasttime = 0;
int16_t firstseq = 0;
struct globalidentifier gid = {};

// convenience method for bswap
#define bswap32(x) \
({ \
    uint32_t __x = (x); \
    ((uint32_t)( \
        (((uint32_t)(__x) & (uint32_t)0x000000ffUL) << 24) | \
        (((uint32_t)(__x) & (uint32_t)0x0000ff00UL) <<  8) | \
        (((uint32_t)(__x) & (uint32_t)0x00ff0000UL) >>  8) | \
        (((uint32_t)(__x) & (uint32_t)0xff000000UL) >> 24) )); \
})

// convenience method for ROL
int16_t rotl16 (int16_t value, unsigned int count) {
    const unsigned int mask = (CHAR_BIT*sizeof(value)-1);
    count &= mask;
    return (value<<count) | (value>>( (-count) & mask ));
}

// the original function (same name) can be found in DevToolsSupport.fr
int8_t * TSGenerateUniqueGlobalID(int8_t *buff) {

    /* Note: the "gid" variable is a reference to the global identifier

    int8_t *buffer = buff;
    if (hasInitialized == 0) {
        hasInitialized = 1;

        pid_t current_pid = getpid();
        gid.pid = (current_pid & 0xff);

        const char *username = [NSUserName() fileSystemRepresentation];
        uint64_t index = 0;
        char letter = username[index];
        int32_t counter = 0;
        int8_t output = 0;

        // perfom the encoding on the username
        do {
            int8_t value = 0x1f;
            letter = username[index];
            if (letter >= 0) {
                value = (int8_t)((letter & 0xff) + packedValueForChar);
            }
```

```
            if (counter != 0) {
                value = (value & 0xff) << counter >> 0x8 | (value & 0xf
            }
            counter = counter + 0x5 & 0x7;
            output = output ^ value;
            index++;
        } while (letter != '\0');
        // store the encoded byte
        gid.user = output;

        int32_t host_id = 0;
        int32_t temp = gethostid(); // this is used even though it is d
        if (temp != -1) {
            host_id = temp;
        }
        // generate the random seed
        int64_t time_seed = [NSDate timeIntervalSinceReferenceDate];
        srandom(((current_pid & 0xff) << 0x10 | host_id) ^ time_seed);
        if (host_id == 0) {
            host_id = random();
        }

        gid.zero = 0;
        gid.host_shift = (host_id >> 0x10) & 0xff; // low byte after sh
        gid.host_h = (host_id & 0xff00) >> 0x8; // high byte
        gid.host_l = (host_id & 0xff);          // low byte
        gid._random = random();
    }

    // increment the random value
    int16_t random_value = gid._random;
    random_value += 1;
    gid._random = random_value;

    // encode the time value and check to make sure we don't have confl
    // (eg when two adds happen in close enough timeframe)
    int64_t time_val = [NSDate timeIntervalSinceReferenceDate];
    int32_t encoded_time = 0;
    if (time_val > lasttime) {
        firstseq = random_value;
        lasttime = time_val;
    }
    else {
        if (firstseq == random_value) {
            lasttime += 1;
        }
    }
    encoded_time = lasttime;
    // now swap byte ordering
    gid._time = bswap32(encoded_time);
```

```
    // rotate the random value for output
    int16_t random_rol = rotl16(random_value, 0x8);
    gid._random = random_rol

    // copy to passed buffer, this will always be 12 bytes
    memcpy(buffer, &gid, sizeof(struct globalidentifier));

    // roll the rolled value again for storing
    gid._random = rotl16(random_rol, 0x8);

    // return passed buffer
    return buffer;
}
```

This function creates a 12 byte identifier, this is converted to a string representation (24 characters), which is what you see in the pbxproj file.
