List of TinyC built-in Functions

- From: TinyC for the SD-8516

For actual built ins there is only putchar() and getchar(). These link to the BASIC IO library at INT $05. After that, a lot can be written in C.

When I say “built-in” I mean these are essentially built-in because I am providing them as a reference and I have validated that they work. They aren't actually built in to V1 *(tinyc.asm)* but you can cut and paste them in front of your main() and then use them. I'll see about including them by default in V2 *(tinyc.c)*.

Assembly, PPU and Audio

There are no graphics or audio commands here, nor is there any way to call an interrupt library, nor is there any way to include assembly. This is because V1 isn't really intended to be a real language. it's supposed to be the languge I write the real C compiler in. But because of various stupid decisions I made during writing it, I am a bit stuck and might need to re-write the assembly version (again).

Instead, I am letting it stew for a while and I might rewrite it, or I might expand it. Or I might just try to force it to work and compile a C version properly.

If I expand it the first thing it will need is access to the PPU and APU/SPU. Until then, the only built-in functions are the ones listed on this page.

print()

```
int print(char *s) {
    int i = 0;
    while (s[i] != 0) {
        putchar(s[i]);
        i = i + 1;
    }
    return 0;
}
```

println()

```
int println(char *s) {
    print(s);
    putchar(13);
    putchar(10);
    return 0;
```

```
}
```

printnum()

```
int printnum(int n) {
    char *buf = 0xDF00;
    int i = 0;
    int neg = 0;
    if (n < 0) {
        neg = 1;
        n = 0 - n;
    }
    if (n == 0) {
        putchar('0');
        return 0;
    }
    while (n > 0) {
        *(buf + i) = n - (n / 10) * 10 + '0';
        n = n / 10;
        i = i + 1;
    }
    if (neg) {
        putchar('-');
    }
    while (i > 0) {
        i = i - 1;
        putchar(buf[i]);
    }
    return 0;
}
```

readline()

```
int readline(char *buf, int max) {
    int i = 0;
    int c = 0;
    while (i < max - 1) {
        c = getchar();
        if (c == 13) {
            putchar(13);
            putchar(10);
            *(buf + i) = 0;
            return i;
        }
        if (c == 8) {
            if (i > 0) {
```

```
                i = i - 1;
                putchar(8);
                putchar(' ');
                putchar(8);
            }
        } else {
            *(buf + i) = c;
            putchar(c);
            i = i + 1;
        }
    }
    *(buf + i) = 0;
    return i;
}
```

Strings stuff

strcmp()

```
int strcmp(char *a, char *b) {
    int i = 0;
    while (a[i] != 0) {
        if (a[i] != b[i]) {
            return a[i] - b[i];
        }
        i = i + 1;
    }
    return a[i] - b[i];
}
```

strlen()

```
int strlen(char *s) {
    int i = 0;
    while (s[i] != 0) {
        i = i + 1;
    }
    return i;
}
```

strcpy()

```
int strcpy(char *dst, char *src) {
```

```
    int i = 0;
    while (src[i] != 0) {
        *(dst + i) = src[i];
        i = i + 1;
    }
    *(dst + i) = 0;
    return i;
}
```

strcat()

```
int strcat(char *dst, char *src) {
    int di = strlen(dst);
    int si = 0;
    while (src[si] != 0) {
        *(dst + di) = src[si];
        di = di + 1;
        si = si + 1;
    }
    *(dst + di) = 0;
    return di;
}
```

strncmp()

```
int strncmp(char *a, char *b, int n) {
    int i = 0;
    while (i < n) {
        if (a[i] != b[i]) {
            return a[i] - b[i];
        }
        if (a[i] == 0) { return 0; }
        i = i + 1;
    }
    return 0;
}
```

strchr()

```
int strchr(char *s, int c) {
    int i = 0;
    while (s[i] != 0) {
        if (s[i] == c) { return i; }
```

```
        i = i + 1;
    }
    return 0 - 1;
}
```

atoi()

```
int atoi(char *s) {
    int n = 0;
    int i = 0;
    while (isdigit(s[i])) {
        n = n * 10 + s[i] - '0';
        i = i + 1;
    }
    return n;
}
```

itoa()

```
int itoa(int n, char *buf) {
    int i = 0;
    int neg = 0;
    if (n < 0) {
        neg = 1;
        n = 0 - n;
    }
    if (n == 0) {
        *(buf) = '0';
        *(buf + 1) = 0;
        return 1;
    }
    char *tmp = 0xDEF0;
    while (n > 0) {
        *(tmp + i) = n - (n / 10) * 10 + '0';
        n = n / 10;
        i = i + 1;
    }
    int j = 0;
    if (neg) {
        *(buf) = '-';
        j = 1;
    }
    while (i > 0) {
        i = i - 1;
        *(buf + j) = tmp[i];
        j = j + 1;
```

```
    }
    *(buf + j) = 0;
    return j;
}
```

Memory stuff

memset()

```
int memset(char *p, int val, int len) {
    int i = 0;
    while (i < len) {
        *(p + i) = val;
        i = i + 1;
    }
    return 0;
}
```

memcpy()

```
int memcpy(char *dst, char *src, int len) {
    int i = 0;
    while (i < len) {
        *(dst + i) = src[i];
        i = i + 1;
    }
    return 0;
}
```

Char stuff

isdigit()

```
int isdigit(int c) {
    if (c >= '0') {
        if (c <= '9') {
            return 1;
        }
    }
    return 0;
```

```
}
```

isalpha()

```
int isalpha(int c) {
    if (c >= 'a') {
        if (c <= 'z') {
            return 1;
        }
    }
    if (c >= 'A') {
        if (c <= 'Z') {
            return 1;
        }
    }
    return 0;
}
```

isspace()

```
int isspace(int c) {
    if (c == ' ') { return 1; }
    if (c == 9) { return 1; }
    if (c == 10) { return 1; }
    if (c == 13) { return 1; }
    return 0;
}
```

isupper()

```
int isupper(int c) {
    if (c >= 'A') {
        if (c <= 'Z') {
            return 1;
        }
    }
    return 0;
}
```

islower()

```
int islower(int c) {
    if (c >= 'a') {
        if (c <= 'z') {
            return 1;
        }
    }
    return 0;
}
```

toupper()

```
int toupper(int c) {
    if (islower(c)) {
        return c - 32;
    }
    return c;
}
```

tolower()

```
int tolower(int c) {
    if (isupper(c)) {
        return c + 32;
    }
    return c;
}
```

Math stuff

abs()

```
int abs(int n) {
    if (n < 0) { return 0 - n; }
    return n;
}
```

min()

```
int min(int a, int b) {
```

```
    if (a < b) { return a; }
    return b;
}
```

max()

```
int max(int a, int b) {
    if (a > b) { return a; }
    return b;
}
```

From:
https://www.appledog.ca/wiki/ - **Appledog**

Permanent link:
https://www.appledog.ca/wiki/doku.php?id=sd:list_of_tinyc_built-in_functions

Last update: **2026/04/30 14:15**