＜ XC8_C の主な標準ライブラリ関数 ＞

【文字列操作関数】 ctype.h

※	関数名	説明	書式
2.1	**isalnum**	文字が英数字(A～Z、a～z、0～9)か検査	int isalnum(int c);
2.2	**isalpha**	文字が英文字(A～Z、a～z)か検査	int isalpha(int c);
2.3	**isblank**	文字が空白またはタブか検査	int isblank (int c);
2.4	**iscntrl**	文字が制御文字(0x00～0x1f、0x7f)か検査	int iscntrl(int c);
2.5	**isdigit**	文字が数字(0～9)か検査	int isdigit(int c);
2.6	**isgraph**	文字が空白を除く印字可能文字(0x21～0x7e)か検査	int isgraph (int c);
2.7	**islower**	文字が英小文字(a～z)か検査	int islower (int c);
2.8	**isprint**	文字が印字可能文字(0x20～0x7e)か検査	int isprint (int c);
2.9	**ispunct**	文字が区切り文字(0x21～0x2f、0x3a～0x40、0x5b～0x60、0x7b～0x7e)か検査	nt ispunct (int c);
2.10	**isspace**	文字 c が空白、タブ、復帰、改行、垂直タブ、改頁(0x09～0x0d、0x20)か検査	int isspace (int c);
2.11	**isupper**	文字が英大文字(A～Z)か検査	int isupper (int c);
2.12	**isxdigit**	文字が16進表示文字(0～9、A～F、a～f)か検査	int isxdigit (int c);
2.13	**tolower**	大文字なら、小文字に変換	int tolower (int c);
2.14	**toupper**	小文字なら、大文字に変換	int toupper (int c);

【算術演算関数】 math.h

※	関数名	説明(double/float/long double)	書　式　[doubleの場合]
8.3	**acos/f/f**	逆余弦関数を計算	double acos (double x);
8.6	**acosh/f/l**	逆双曲線余弦関数を計算	double acosh(double x);
8.9	**asin/f/l**	逆正弦関数を計算	double asin (double x);
8.12	**asinh/f/l**	逆双曲線正弦関数を計算	double asinh(double x);
8.15	**atan/f/l**	逆正接関数を計算	double atan (double x);
8.18	**atanh/f/l**	逆双曲線正弦関数を計算	double atanh(double x);
8.21	**atan2/f/l**	y/x の逆正接関数を計算	double atan2 (double y, double x);
8.24	**cbrt/f/l**	立方根を計算	double cbrt(double x);
8.27	**ceil/f/l**	小数点以下切り上げ値を返す	double ceil(double x);
8.30	**copysign/f/l**	x の値を返す。ただし符号はyと同じ	double copysign(double x, double y);
8.33	**cos/f/l**	コサイン(余弦)値を返す	double cos (double x);
8.36	**cosh/f/l**	双曲線余弦関数を計算	double cosh (double x);
8.39	**erf/f/l**	引数の誤差関数を計算	double erf(double x);
8.42	**erfc/f/l**	引数の相補誤差関数を計算	double erfc(double x);
8.45	**eval_poly**	多項式を評価し、その結果を返す	void eval_poly(double x, const double * d, int n);
8.46	**exp/f/l**	指数関数 e の x乗を計算	double exp(double x);
8.49	**expm1/f/l**	x から 1 を引いた指数関数を計算	double expm1(double x);
8.52	**exp2/f/l**	x の 2 を底とする指数関数 を計算	double exp2(double x);
8.55	**fabs/f/l**	絶対値を返す	double fabs(double x);
8.58	**fdim/f/l**	2 つの引数の正の差を計算	double fdim(double x, double y);
8.61	**floor/f/l**	データ x を切り捨てて返す	double floor (double x);
8.64	**fmax/f/l**	大きい方の引数の値を返す	double fmax(double x, double y);
8.67	**fmin/f/l**	小さい方の引数の値を返す	double fmin(double x, double y);
8.70	**fmod/f/l**	x / y の余りを返す	double fmod(double x, double y);
8.73	**fpclassify**	引数を浮動小数点カテゴリとして分類	int isgreater(floating-point x);

8.74	**frexp/f/l**	小数と指数を取得	double frexp (double x, int *exp);
8.77	**hypot/f/l**	平方和の平方根を計算	double hypot(double x, double y);
8.80	**itogb/f/l**	符号付き整数指数を計算	int ilogb(double x);
8.83	**isfinits**	引数が有限である場合、true を返す	int isfinite(floating-point x);
8.84	**isgreater**	最初の引数が2番目より大きいか判断	int isgreater(floating-point x, floating-point y)
8.85	**isgreaterequal**	最初の引数が2番目より大or等しいか判断	int isgreaterequal(floating-point x, floating-point y);
8.86	**isinf**	引数が無限大の場合は true を返します	int isinf(floating-point x);
8.87	**isless**	最初の引数が2番目よりも小さいか判断	int isless(floating-point x, floating-point y);
8.88	**islessequal**	最初の引数が2番目より小or等しいか判断	int islessequal(floating-point x, floating-point y);
8.89	**islessgrater**	最初の引数が2番目よりも小or大か判断	int islessgreater(floating-point x, floating-point y);
8.90	**isnan**	引数が NaN(非数)の時true を返す	int isnan(floating-point x);
8.91	**isnormal**	引数が通常の場合、true を返す	int isnormal(floating-point x);
8.92	**isunordered**	引数が順序付けされていないか判断	int isunordered(floating-point x, floating-point y);
8.93	**ldexp/f/l**	指数 2 を掛けた結果を計算	double ldexp(double x, int ex);
8.96	**lgamma/f/l**	引数のガンマの絶対値の自然対数を計算	double lgamma(double x);
8.99	**lirint/f/l**	最も近い整数値に丸められた引数を返す	long long int llrint(double x);
8.102	**liround/f/l**	整数値に丸められた引数を返す	long long int llround(double x);
8.105	**log/f/l**	自然対数 log を返す	double log(double x);
8.108	**log10/f/l**	常用対数 log10 を返す	double log10(double x);
8.111	**log1p/f/l**	引数に 1 を加えた値の自然対数を計算	double log1p(double x);
8.114	**log2/f/l**	2 を底とする対数を計算	double log2(double x);
8.117	**logb/f/l**	符号付き指数を計算	double logb(double x);
8.120	**lrint/f/l**	最も近い整数値に丸めらた引数を返す	long int lrint(double x);
8.123	**lround/f/l**	整数値に丸められた引数を返す	long int lround(double x);
8.126	**modf/f/l**	小数部分と整数部分に分割する	double modf(double x, double *pint);
8.129	**nan/f/l**	Quietな NaN（非数）を返す	double nan(const char * tagp);
8.132	**nearbyint/f/l**	整数値に丸められた引数を返す	double nearbyint(double x);
8.135	**nextafter/f/l**	関数で表示可能なy方向のxの次の値	double nextafter(double x, double y);
8.138	**nextloward/f/l**	関数で表示可能なy方向のxの次の値	double nexttoward(double x, long double y);
8.141	**pow/f/l**	x の y乗を返す	double pow(double x, double y);
8.144	**remainder/f/l**	x REM y を計算	double remainder(double x, double y);
8.147	**remquo/f/l**	x REM y を計算	double remquo(double x, double y, int * quo);
8.150	**rint/f/l**	整数値に丸められた引数を返す	double rint(double x);
8.153	**round/f/l**	整数値に丸められた引数を返す	double round(double x);
8.156	**scalbn/f/l**	符号付き指数を計算	double scalbn(double x, int n);
8.159	**scalbin/f/l**	符号付き指数を計算	double scalbln(double x, long int n);
8.162	**signbit**	引数が負の場合は true を返す	int signbit(floating-point x);
8.163	**sin/f/l**	サイン値の返す	double sin (double x);
8.166	**sinh/f/l**	双曲線正弦関数を計算	double sinh (double x);
8.169	**sqrt/f/l**	x の平方根を返す	double sqrt(double x);
8.172	**tan/f/l**	正接値の返す	double tan (double x);
8.175	**tanh/f/l**	双曲線正接関数を計算	double tanh(double x);
8.178	**tgamma/f/l**	引数のガンマ関数を計算	double tgamma(double x);
8.181	**trunc/f/l**	引数値以下の整数値に丸める	double trunc(double x);

【入出力関数】　　stdlib.h

※	関数名	説明	書式
14.1	**getchar**	標準入力から1文字を取得	int getchar(void);
14.2	**gets**	標準入力から1行文字列を取得	char *gets(char *s);
14.3	**perror**	システムエラーメッセージの出力	void perror(const char * s);
14.4	**printf**	標準出力へ書式付で出力	int printf(const char *format, ...);
14.5	**putc**	ファイルへ1文字出力(マクロ)	int putc(int c, FILE *stream);
14.6	**putch**	標準出力へ1文字出力	void putch(char c);
14.7	**putchar**	標準出力へ1文字出力	int putchar(int c);
14.8	**puts**	標準出力へ1行文字列を出力	int puts(const char *s);
14.9	**scanf**	標準入力から書式付で入力	int scanf(const char *format, ...);
14.10	**sprintf**	書式指定変換した出力を文字列に格納	int sprintf(char *s, const char *format, ...);
14.11	**sscanf**	文字列から書式指定に従い入力	int sscanf(const char *s, const char *format, ...);

【一般的な関数】　　stdlib.h

※	関数名	説明	書式
15.1	**abort**	現在のプロセスを中止し、リセットされる	void abort(void);
15.2	**abs**	int型データ n の絶対値を int型で返却	int abs(int i);
15.3	**atexit**	プログラム正常終了時に呼び出す関数を登録	int atexit(void(*func)(void));
15.4	**atof**	文字列を double型変数に変換して返却	double atof(const char *s);
15.5	**atoi**	文字列を int型変数に変換して返却	int atoi(const char *s);
15.6	**atol**	文字列を long型変数に変換して返却	long atol(const char *s);
15.7	**atoll**	文字列を long-long型変数に変換して返却	long long atol(const char *s);
15.8	**bsearch**	二分探索を実行	void *bsearch(const void *key, const void *base, size_t nelem, size_t size, int (*cmp) (const void *ck, const void *ce));
15.9	**div**	2 つの数値の商と剰余を計算	div_t div(int numer, int denom);
15.10	**exit**	クリーンアップ後にプログラム終了。	void exit(int status);
15.11	**labs**	long整数の絶対値を計算	long labs(long i);
15.12	**llabs**	long-long整数の絶対値を計算	long long llabs(long long i);
15.13	**ldiv**	2 つのlong整数の商と剰余を計算	ldiv_t ldiv(long numer, long denom);
15.14	**lldiv**	2 つのlong-long整数の商と剰余を計算	ldiv_t lldiv(long numer, long denom);
15.15	**qsort**	オブジェクトの並べ替えと配列	void qsort(void *base, size_t nmemb, size_t size, int (*compar)(const void *, const void *));
15.16	**rand**	整数型の疑似乱数を生成	int rand(void);
15.17	**srand**	以降の rand 関数で使用されるシードを指定	int srand(unsigned int seed);
15.18	**strtod**	文字列をdouble値に変換(エラー検出あり)	double strtod(const char * restrict nptr, char ** restrict endptr);
15.19	**strtof**	文字列を float 浮動小数点値に変換	float strtod(const char * restrict nptr, char ** restrict endptr);
15.20	**dtrtold**	文字列を long double 浮動小数点値に変換	long double strtod(const char * restrict nptr, char ** restrict endptr);
15.21	**strtol**	文字列をlong値に変換(エラー検出、基数指定	long int strtol(const char * restrict nptr, char ** restrict endptr, int base);
15.22	**strtoll**	文字列を長整数値に変換	long long int strtoll(const char * restrict nptr, char ** restrict endptr, int base);
15.23	**strtoul**	文字列をunsigned long値に変換(エラー検出、基数指定あり)	unsigned long int strtol(const char * restrict nptr, char ** restrict endptr, int base);
15.24	**strtoull**	文字列を unsigned long long 整数値に変換	unsigned long long int strtoull(const char * restrict nptr, char ** restrict endptr, int base);

【文字列処理関数】 string.h

※	関数名	説明	書式
16.1	**memchr**	文字を n バイト中から検索する	void *memchr(const void *s, int c, size_t n);
16.2	**memcmp**	n バイトメモリブロックの比較	int memcmp(const void *s1, const void *s2, size_t n);
16.3	**memcpy**	n バイトメモリブロックのコピー	void *memcpy(void *dst, const void *src, size_t n);
16.4	**memmove**	n バイトメモリブロックの移動	void *memmove(void *s1, const void *s2, size_t n);
16.5	**memset**	n バイトメモリブロックのセット	void *memset(void *s, int c, size_t n);
16.6	**strcat**	文字列の連結	char *strcat(char *s1, const char *s2);
16.7	**strchr**	文字列の先頭から文字を検索する	char *strchr(const char *s, int c);
16.8	**strcmp**	文字列の比較	int strcmp(const char *s1, const char *s2);
16.9	**strcoll**	ある文字列を別の文字列と比較	int strcoll(const char *s1, const char *s2);
16.10	**strcpy**	文字列のコピー	char *strcpy(char *s1, const char *s2);
16.11	**strcspn**	文字列から文字群が含まれない長さ	size_t strcspn(const char *s1, const char *s2);
16.12	**strerror**	エラー番号対応エラーメッセージ取得	char *strerror(int errcode);
16.13	**strlen**	文字列の長さの取得	size_t strlen(const char *s);
16.14	**strncat**	文字列を n 文字連結	char *strncat(char *s1, const char *s2, size_t n);
16.15	**strncmp**	文字列を n 文字比較	int strncmp(const char *s1, const char *s2, size_t n);
16.16	**strncpy**	文字列を n 文字コピー	char *strncpy(char *s1, const char *s2, size_t n);
16.17	**strpbrk**	文字列から文字群が見つかった位置	char *strpbrk(const char *s1, const char *s2);
16.18	**strrchr**	文字列の最後から文字を検索する	char *strrchr(const char *s, int c);
16.19	**strspn**	文字列から文字群が含まれる長さ	size_t strspn(const char *s1, const char *s2);
16.20	**strstr**	文字列1から文字列2を検索	char *strstr(const char *s1, const char *s2);
16.21	**strtok**	文字列を区切り文字で分解する	char *strtok(char *s1, const char *s2);
16.22	**strxfrm**	ロケール依存規則により文字列を変換	size_t strxfrm(char *s1, const char *s2, size_t n);

【時刻・日付管理関数】 time.h

※	関数名	説明	書式
17.1	**asctime**	tm構造体を文字列へ変換	char *asctime(const struct tm *tptr);
17.2	**ctime**	日付と時刻を文字列へ変換	char *ctime(const time_t *tod);
17.3	**difftime**	2つの時間の差(秒単位)を算出	double difftime(time_t t1, time_t t0);
17.4	**gmtime**	tm構造体(グリニッジ時間)への変換	struct tm *gmtime(const time_t *tod);
17.5	**localtime**	tm構造体(日本時間)への変換	struct tm *localtime(const time_t *tod);
17.6	**mktime**	tm構造体を time_t型に変換	time_t mktime(struct tm *tptr);
17.7	**strftime**	formatに基づきtm構造を文字列に変換	size_t strftime(char *s, size_t n, const char *format, const struct tm *tptr);
17.8	**time**	現在時刻の取得	time_t time(time_t *tod);

【デバイス固有関数】 xc.h

※	関数名	説明	書式
18.1	**CLRWDT**	ウオッチドグタイマーをクリアする	void CLRWDT(void);
18.2	**di**	割り込みを禁止する(GIE = 0)	void di(void);
18.3	**ei**	割り込みを許可する(GIE = 1)	void ei(void);
18.14	**__delay_ms**	実行を遅らせる組み込み関数(ms単位)	void __delay_ms(unsigned long time);
18.16	**__delay_us**	実行を遅らせる組み込み関数(μs単位)	void __delay_us(unsigned long time);
18.23	**SLEEP**	デバイスをスリープ モードにする	void SLEEP(void);