コンパイラパッケージ

RJJ10J2803-0100
Rev.1.00
2010.07.30

アプリケーションノート：＜ STL ライブラリ V.1.00.00 ＞

サポート機能一覧

本ドキュメントでは、Renesas C/C++コンパイラ向けSTLライブラリ V.1.00.00の使用方法を説明します。

目次

1. STL ライブラリサポートヘッダー一覧

本資料は、以下のルネサス エレクトロニクス社（以下 弊社）製 C/C++コンパイラ向けに提供するSTLライブラリ V.1.00.00(以下 本STLライブラリ)でサポートしている機能を記載しています。

- RX ファミリ用 C/C++コンパイラ V.1.00.00 以降
- SuperH ファミリ用 C/C++コンパイラ V.9.04.00 以降

本 STL ライブラリでサポートしているヘッダーは以下の通りです。

	ヘッダー名	メソッド有無	備考
1	<algorithm>	○	-
2	<bitset>	○	-
3	<complex>	○	-
4	<deque>	○	-
5	<exception>	○	-
6	<functional>	○	-
7	<iterator>	○	-
8	<limits>	○	-
9	<list>	○	-
10	<map>	○	-
11	<memory>	○	-
12	<numeric>	○	-
13	<queue>	○	-
14	<set>	○	-
15	<stack>	○	-
16	<stdexcept>	○	-
17	<stlstring>	○	通常<string>ですが、EC++との競合を防ぐ為<stlstring>としています。
18	<typeinfo>	○	-
19	<utility>	○	-
20	<valarray>	○	-
21	<vector>	○	-
22	<cassert>	×	<assert.h>の参照
23	<cctyoe>	×	<ctype.h>の参照
24	<cerrno>	×	<errno.h>の参照
25	<cfloat>	×	<float.h>の参照
26	<climits>	×	<limits.h>の参照
27	<cmath>	×	<math.h>の参照
28	<csetjmp>	×	<setjmp.h>の参照
29	<cstdarg>	×	<stdarag.h>の参照
30	<cstddef>	×	<stddef.h>の参照
31	<cstdio>	×	<stdio.h>の参照
32	<cstdlib>	×	<stdlib.h>の参照
33	<cstring>	×	<string.h>の参照
34	<new.h>	×	<new>の参照

以降は、各ヘッダーがサポートしているメソッドを記載します。

2. <algorithm>

<algorithm>でサポートしているメソッドは以下になります。

algorithm クラスのメソッド一覧	
1	template <class _Tp> void swap(_Tp& _a, _Tp& _b)
2	void iter_swap(_ForwardIter1 _i1, _ForwardIter2 _i2)
3	template <class _Tp> inline const _Tp& (min)(const _Tp& _a, const _Tp& _b)
4	template <class _Tp> inline const _Tp& (min)(const _Tp& _a, const _Tp& _b, _Compare _comp)
5	template <class _Tp> inline const _Tp& (max)(const _Tp& _a, const _Tp& _b)
6	template <class _Tp> inline const _Tp& (max)(const _Tp& _a, const _Tp& _b, _Compare _comp)
7	_OutputIter copy(_InputIter _first, _InputIter _last, _OutputIter _result)
8	_OutputIter copy_backward(_InputIter _first, _InputIter _last, _OutputIter _result)
9	template <class _ForwardIter, class _Tp> void fill(_ForwardIter _first, _ForwardIter _last, const _Tp& _val)
10	void fill(unsigned char* _first, unsigned char* _last, const unsigned char& _val)
11	void fill(signed char* _first, signed char* _last,const signed char& _val)
12	void fill(char* _first, char* _last, const char& _val)
13	template <class _OutputIter, class _Size, class _Tp> void fill_n(_OutputIter _first, _Size _n, const _Tp& _val)
14	template <class _InputIter1, class _InputIter2> ::stlpmtx_std::pair<_InputIter1, _InputIter2> mismatch(_InputIter1 _first1, _InputIter1 _last1, _InputIter2 _first2)
15	template <class _InputIter1, class _InputIter2, class _BinaryPredicate> ::stlpmtx_std::pair<_InputIter1, _InputIter2> mismatch(_InputIter1 _first1,_InputIter1 _last1,_InputIter2 _first2,_BinaryPredicate _binary_pred)
16	template <class _InputIter1, class _InputIter2> bool equal(_InputIter1 _first1, _InputIter1 _last1, _InputIter2 _first2)
17	template <class _InputIter1, class _InputIter2, class _BinaryPredicate> bool equal(_InputIter1 _first1, _InputIter1 _last1, _InputIter2 _first2, _BinaryPredicate _binary_pred)
18	template <class _InputIter1, class _InputIter2> bool lexicographical_compare(_InputIter1 _first1, _InputIter1 _last1,_InputIter2 _first2, _InputIter2 _last2);
19	template <class _InputIter1, class _InputIter2, class _Compare> bool lexicographical_compare(_InputIter1 _first1, _InputIter1 _last1,_InputIter2 _first2, _InputIter2 _last2,_Compare _comp);
20	bool lexicographical_compare(const unsigned char* _first1,const unsigned char* _last1,const unsigned char* _first2,const unsigned char* _last2)
21	int lexicographical_compare_3way(_InputIter1 _first1, _InputIter1 _last1,_InputIter2 _first2, _InputIter2 _last2);
22	template <class _InputIter, class _Tp> iterator_traits<_InputIter> ::difference_type count(_InputIter _first, _InputIter _last, const _Tp& _val)
23	template <class _InputIter, class _Tp> _InputIter find(_InputIter _first, _InputIter _last, const _Tp& _val)
24	_InputIter find_if(_InputIter _first, _InputIter _last, _Predicate _pred)
25	_ForwardIter1 search(_ForwardIter1 _first1, _ForwardIter1 _last1, _ForwardIter2 _first2, _ForwardIter2 _last2, _BinaryPred _predicate);
26	template <class _ForwardIter1, class _ForwardIter2,class _BinaryPredicate> _ForwardIter1 find_end(_ForwardIter1 _first1, _ForwardIter1 _last1, _ForwardIter2 _first2, _ForwardIter2 _last2, _BinaryPredicate _comp);
27	template <class _ForwardIter, class _Tp> void replace(_ForwardIter _first, _ForwardIter _last, const _Tp& _old_value, const _Tp& _new_value)
28	template <class _InputIter, class _Function> nline _Function for_each(_InputIter _first, _InputIter _last, _Function _f)
29	template <class _ForwardIter, class _BinaryPredicate> _ForwardIter adjacent_find(_ForwardIter _first, _ForwardIter _last, _BinaryPredicate _binary_pred)
30	template <class _ForwardIter> _ForwardIter adjacent_findadjacent_find(_ForwardIter _first, _ForwardIter _last)
31	template <class _InputIter, class _Tp, class _Size> _STLP_INLINE_LOOP void count(_InputIter _first, _InputIter _last, const _Tp& _val, _Size& _n)

32	template <class _InputIter, class _Predicate> _STLP_INLINE_LOOP _STLP_DIFFERENCE_TYPE(_InputIter) count_if(_InputIter __first, _InputIter __last, _Predicate __pred)
33	template <class _InputIter, class _Predicate, class _Size> _STLP_INLINE_LOOP void count_if(_InputIter __first, _InputIter __last, _Predicate __pred, _Size& __n)
34	template <class _ForwardIter1, class _ForwardIter2> _ForwardIter1 search(_ForwardIter1 __first1, _ForwardIter1 __last1,_ForwardIter2 __first2, _ForwardIter2 __last2);
35	template <class _ForwardIter, class _Integer, class _Tp> _ForwardIter search_n(_ForwardIter __first, _ForwardIter __last, _Integer __count, const _Tp& __val);
36	template <class _ForwardIter, class _Integer, class _Tp, class _BinaryPred> _ForwardIter search_n(_ForwardIter __first, _ForwardIter __last, _Integer __count, const _Tp& __val, _BinaryPred __binary_pred);
37	template <class _InputIter, class _ForwardIter> inline _InputIter find_first_of(_InputIter __first1, _InputIter __last1,_ForwardIter __first2, _ForwardIter __last2)
38	template <class _InputIter, class _ForwardIter, class _BinaryPredicate> find_first_of(_InputIter __first1, _InputIter __last1,
39	template <class _ForwardIter1, class _ForwardIter2> _ForwardIter1 find_end(_ForwardIter1 __first1, _ForwardIter1 __last1, _ForwardIter2 __first2, _ForwardIter2 __last2);
40	template <class _ForwardIter1, class _ForwardIter2> _STLP_INLINE_LOOP _ForwardIter2 swap_ranges(_ForwardIter1 __first1, _ForwardIter1 __last1, _ForwardIter2 __first2)
41	template <class _InputIter, class _OutputIter, class _UnaryOperation> _OutputIter transform(_InputIter __first, _InputIter __last, _OutputIter __result, _UnaryOperation __opr)
42	template <class _InputIter1, class _InputIter2, class _OutputIter, class _BinaryOperation> _OutputIter transform(_InputIter1 __first1, _InputIter1 __last1, _InputIter2 __first2, _OutputIter __result,_BinaryOperation __binary_op)
43	template <class _ForwardIter, class _Predicate, class _Tp> _ForwardIter remove_if(_ForwardIter __first, _ForwardIter __last, _Predicate __pred)
44	template <class _InputIter, class _OutputIter, class _Tp> _OutputIter replace_copy(_InputIter __first, _InputIter __last,_OutputIter __result, const _Tp& __old_value, const _Tp& __new_value)
45	template <class _Iterator, class _OutputIter, class _Predicate, class _Tp> _OutputIter replace_copy_if(_Iterator __first, _Iterator __last, _OutputIter __result, _Predicate __pred, const _Tp& __new_value)
46	template <class _ForwardIter, class _Generator> void generate(_ForwardIter __first, _ForwardIter __last, _Generator __gen)
47	template <class _OutputIter, class _Size, class _Generator> void generate_n(_OutputIter __first, _Size __n, _Generator __gen)
48	template <class _InputIter, class _OutputIter, class _Tp> _OutputIter remove_copy(_InputIter __first, _InputIter __last,_OutputIter __result, const _Tp& __val)
49	template <class _InputIter, class _OutputIter, class _Predicate> _OutputIter remove_copy_if(_InputIter __first, _InputIter __last, _OutputIter __result, _Predicate __pred)
50	template <class _ForwardIter, class _Tp> _ForwardIter remove(_ForwardIter __first, _ForwardIter __last, const _Tp& __val)
51	template <class _ForwardIter, class _Predicate> void replace_if(_ForwardIter __first, _ForwardIter __last, _Predicate __pred, const _Tp& __new_value)
52	template <class _InputIter, class _OutputIter> _OutputIter unique_copy(_InputIter __first, _InputIter __last, _OutputIter __result)
53	template <class _InputIter, class _OutputIter, class _BinaryPredicate> _OutputIter unique_copy(_InputIter __first, _InputIter __last,_OutputIter __result, _BinaryPredicate __binary_pred)
54	template <class _ForwardIter, class _BinaryPredicate> _ForwardIter unique(_ForwardIter __first, _ForwardIter __last, _BinaryPredicate __binary_pred)
55	template <class _BidirectionalIter> void reverse(_BidirectionalIter __first, _BidirectionalIter __last)
56	template <class _BidirectionalIter, class _OutputIter> _OutputIter reverse_copy(_BidirectionalIter __first, _BidirectionalIter __last, _OutputIter __result)
57	template <class _ForwardIter> void rotate(_ForwardIter __first, _ForwardIter __middle, _ForwardIter __last)
58	template <class _ForwardIter, class _OutputIter> _OutputIter rotate_copy(_ForwardIter __first, _ForwardIter __middle,_ForwardIter __last, _OutputIter __result)
59	template <class _RandomAccessIter> void random_shuffle(_RandomAccessIter __first, _RandomAccessIter __last)
60	template <class _RandomAccessIter, class _RandomNumberGenerator> void random_shuffle(_RandomAccessIter __first, _RandomAccessIter __last, _RandomNumberGenerator& __rand)
61	template <class _ForwardIter, class _OutputIter, class _Distance> OutputIter random_sample_n(_ForwardIter __first, _ForwardIter __last, _OutputIter __out_ite, const _Distance __n)

62	template <class _ForwardIter, class _OutputIter, class _Distance, class _RandomNumberGenerator> _OutputIter random_sample_n(_ForwardIter _first, _ForwardIter _last, _OutputIter _out_ite, const _Distance _n, _RandomNumberGenerator& _rand)
63	template <class _InputIter, class _RandomAccessIter> _RandomAccessIter random_sample(_InputIter _first, _InputIter _last, _RandomAccessIter _out_first, _RandomAccessIter _out_last)
64	template <class _InputIter, class _RandomAccessIter, class _RandomNumberGenerator> _RandomAccessIter random_sample(_InputIter _first, _InputIter _last, _RandomAccessIter _out_first, _RandomAccessIter _out_last, _RandomNumberGenerator& _rand)
65	template <class _ForwardIter, class _Predicate> _ForwardIter partition(_ForwardIter _first, _ForwardIter _last, _Predicate __pred)
66	template <class _ForwardIter, class _Predicate> _ForwardIter stable_partition(_ForwardIter _first, _ForwardIter _last, _Predicate _pred)
67	template <class _RandomAccessIter> void sort(_RandomAccessIter _first, _RandomAccessIter _last)
68	template <class _RandomAccessIter, class _Compare> void sort(_RandomAccessIter _first, _RandomAccessIter _last, _Compare _comp)
69	template <class _RandomAccessIter> void stable_sort(_RandomAccessIter _first,_RandomAccessIter _last)
70	template <class _RandomAccessIter, class _Compare> void stable_sort(_RandomAccessIter _first, _RandomAccessIter _last, _Compare _comp)
71	template <class _RandomAccessIter> void partial_sort(_RandomAccessIter _first, _RandomAccessIter _middle, _RandomAccessIter _last)
72	template <class _RandomAccessIter, class _Compare> void partial_sort(_RandomAccessIter _first,_RandomAccessIter _middle, _RandomAccessIter _last, _Compare _comp)
73	template <class _InputIter, class _RandomAccessIter> _RandomAccessIter partial_sort_copy(_InputIter _first, _InputIter _last, _RandomAccessIter _result_first, _RandomAccessIter _result_last)
74	template <class _InputIter, class _RandomAccessIter, class _Compare> partial_sort_copy(_InputIter _first, _InputIter _last, _RandomAccessIter _result_first, _RandomAccessIter _result_last, _Compare _comp)
75	template <class _RandomAccessIter> void nth_element(_RandomAccessIter _first, _RandomAccessIter _nth, _RandomAccessIter _last)
76	template <class _RandomAccessIter, class _Compare> void nth_element(_RandomAccessIter _first, _RandomAccessIter _nth, _RandomAccessIter _last, _Compare _comp)
77	template <class _ForwardIter, class _Tp> _ForwardIter lower_bound(_ForwardIter _first, _ForwardIter _last, const _Tp& _val)
78	template <class _ForwardIter, class _Tp, class _Compare> _ForwardIter lower_bound(_ForwardIter _first, _ForwardIter _last, const _Tp& _val, _Compare _comp)
79	template <class _ForwardIter, class _Tp> inline _ForwardIter upper_bound(_ForwardIter _first, _ForwardIter _last, const _Tp& _val)
80	template <class _ForwardIter, class _Tp, class _Compare> inline _ForwardIter upper_bound(_ForwardIter _first, _ForwardIter _last,const _Tp& _val, _Compare _comp)
81	template <class _ForwardIter, class _Tp> inline pair<_ForwardIter, _ForwardIter> equal_range(_ForwardIter _first, _ForwardIter _last, const _Tp& _val)
82	template <class _ForwardIter, class _Tp, class _Compare> inline pair<_ForwardIter, _ForwardIter> equal_range(_ForwardIter _first, _ForwardIter _last, const _Tp& _val, _Compare _comp)
83	template <class _ForwardIter, class _Tp> bool binary_search(_ForwardIter _first, _ForwardIter _last, const _Tp& _val)
84	template <class _ForwardIter, class _Tp, class _Compare> bool binary_search(_ForwardIter _first, _ForwardIter _last, const _Tp& _val, _Compare _comp)
85	template <class _InputIter1, class _InputIter2, class _OutputIter> _OutputIter merge(_InputIter1 _first1, _InputIter1 _last1, _InputIter2 _first2, _InputIter2 _last2, _OutputIter _result)
86	template <class _InputIter1, class _InputIter2, class _OutputIter, class _Compare> _OutputIter merge(_InputIter1 _first1, _InputIter1 _last1, _InputIter2 _first2, _InputIter2 _last2, OutputIter _result, _Compare _comp)
87	template <class _BidirectionalIter> void inplace_merge(_BidirectionalIter _first, _BidirectionalIter _middle, _BidirectionalIter _last) ;
88	template <class _BidirectionalIter, class _Compare> void inplace_merge(_BidirectionalIter _first, _BidirectionalIter _middle, _BidirectionalIter _last, _Compare _comp)
89	template <class _InputIter1, class _InputIter2> bool includes(_InputIter1 _first1, _InputIter1 _last1, _InputIter2 _first2, _InputIter2 _last2)
90	template <class _InputIter1, class _InputIter2, class _Compare> bool includes(_InputIter1 _first1, _InputIter1 _last1, _InputIter2 _first2, _InputIter2 _last2, _Compare _comp)

91	template <class _InputIter1, class _InputIter2, class _OutputIter> OutputIter set_union(_InputIter1 _first1, _InputIter1 _last1, _InputIter2 _first2, _InputIter2 _last2, _OutputIter _result)
92	template <class _InputIter1, class _InputIter2, class _OutputIter, class _Compare> _OutputIter set_union(_InputIter1 _first1, _InputIter1 _last1, _InputIter2 _first2, _InputIter2 _last2, _OutputIter _result, _Compare _comp)
93	template <class _InputIter1, class _InputIter2, class _OutputIter> utputIter set_intersection(_InputIter1 _first1, _InputIter1 _last1,_InputIter2 _first2, _InputIter2 _last2, _OutputIter _result)
94	template <class _InputIter1, class _InputIter2, class _OutputIter, class _Compare> _OutputIter set_intersection(_InputIter1 _first1, _InputIter1 _last1, _InputIter2 _first2, _InputIter2 _last2, _OutputIter _result, _Compare _comp)
95	template <class _InputIter1, class _InputIter2, class _OutputIter> _OutputIter set_difference(_InputIter1 _first1, _InputIter1 _last1, _InputIter2 _first2, _InputIter2 _last2, _OutputIter _result)
96	template <class _InputIter1, class _InputIter2, class _OutputIter, class _Compare> _OutputIter set_difference(_InputIter1 _first1, _InputIter1 _last1, _InputIter2 _first2, _InputIter2 _last2, _OutputIter _result, _Compare _comp)
97	template <class _InputIter1, class _InputIter2, class _OutputIter,class _Compare> _OutputIter set_symmetric_difference(_InputIter1 _first1, _InputIter1 _last1, _InputIter2 _first2, _InputIter2 _last2, _OutputIter _result)
98	template <class _InputIter1, class _InputIter2, class _OutputIter> _OutputIter set_symmetric_difference(_InputIter1 _first1, _InputIter1 _last1, _InputIter2 _first2, _InputIter2 _last2, _OutputIter _result, Compare _comp)
99	template <class _ForwardIter> _ForwardIter max_element(_ForwardIter _first, _ForwardIter _last);
100	template <class _ForwardIter, class _Compare> _ForwardIter max_element(_ForwardIter _first, _ForwardIter _last,_Compare _comp);
101	template <class _ForwardIter> _ForwardIter min_element(_ForwardIter _first, _ForwardIter _last);
102	template <class _ForwardIter, class _Compare> _ForwardIter min_element(_ForwardIter _first, _ForwardIter _last,_Compare _comp);
103	template <class _BidirectionalIter> bool next_permutation(_BidirectionalIter _first, _BidirectionalIter _last);
104	template <class _BidirectionalIter, class _Compare> bool next_permutation(_BidirectionalIter _first, _BidirectionalIter _last, _Compare _comp);
105	template <class _BidirectionalIter> bool prev_permutation(_BidirectionalIter _first, _BidirectionalIter _last);
106	template <class _BidirectionalIter, class _Compare> bool prev_permutation(_BidirectionalIter _first, _BidirectionalIter _last, _Compare _comp);
107	template <class _RandomAccessIter> bool is_heap(_RandomAccessIter _first, _RandomAccessIter _last)
108	template <class _RandomAccessIter, class _StrictWeakOrdering> bool is_heap(_RandomAccessIter _first, _RandomAccessIter _last, _StrictWeakOrdering _comp)
109	bool is_sorted(_ForwardIter _first, _ForwardIter _last)
110	bool is_sorted(_ForwardIter _first, _ForwardIter _last, _StrictWeakOrdering _comp)

3. <bitset>

<bitset>でサポートしているメソッドは以下になります。

bitset クラスのメソッド一覧	
1	reference& operator=(bool _x)
2	reference& operator=(const reference& _j)
3	bool operator~()
4	operator bool()
5	reference& flip()
6	bitset()
7	bitset(unsigned long _val)
8	bitset(const basic_string<_CharT,_Traits,_Alloc>& _s, size_t _pos = 0)
9	bitset(const basic_string<_CharT, _Traits, _Alloc>& _s, size_t _pos, size_t _n)
10	bitset<_Nb>& operator&=(const bitset<_Nb>& _rhs)
11	bitset<_Nb>& operator|=(const bitset<_Nb>& _rhs)
12	bitset<_Nb>& operator^=(const bitset<_Nb>& _rhs)
13	bitset<_Nb>& operator<<=(size_t _pos)
14	bitset<_Nb>& operator>>=(size_t _pos)
15	bitset<_Nb>& _Unchecked_set(size_t _pos)
16	bitset<_Nb>& _Unchecked_set(size_t _pos, int _val)
17	bitset<_Nb>& _Unchecked_reset(size_t _pos)
18	bitset<_Nb>& _Unchecked_flip(size_t _pos)
19	bool _Unchecked_test(size_t _pos)
20	bitset<_Nb>& set()
21	bitset<_Nb>& set(size_t _pos)
22	bitset<_Nb>& set(size_t _pos, int _val)
23	bitset<_Nb>& reset()
24	bitset<_Nb>& reset(size_t _pos)
25	bitset<_Nb>& flip()
26	bitset<_Nb>& flip(size_t _pos)
27	bitset<_Nb> operator~()
28	reference operator[](size_t _pos)
29	bool operator[](size_t _pos)
30	unsigned long to_ulong()
31	size_t count()
32	size_t size()
33	bool operator==(const bitset<_Nb>& _rhs)
34	bool operator!=(const bitset<_Nb>& _rhs)
35	bool test(size_t _pos)
36	bool any()
37	bool none()
38	bitset<_Nb> operator<<(size_t _pos)
39	bitset<_Nb> operator>>(size_t _pos)
40	size_t _Find_first() const
41	size_t _Find_next(size_t _prev) const
42	void _M_copy_from_string(const basic_string<_CharT,_Traits,_Alloc>& _s, size_t _pos, size_t _n)
43	void _M_copy_to_string(basic_string<_CharT, _Traits, _Alloc>& _s) const
44	bitset<_Nb> operator&(const bitset<_Nb>& _x, const bitset<_Nb>& _y)
45	bitset<_Nb> operator|(const bitset<_Nb>& _x, const bitset<_Nb>& _y)
46	bitset<_Nb> operator^(const bitset<_Nb>& _x, const bitset<_Nb>& _y)

4. <complex>

<complex>でサポートしているメソッドは以下になります。

comlex クラスのメソッド一覧	
1	complex()
2	complex(const value_type& _x)
3	complex(const value_type& _x, const value_type& _y)
4	complex(const _Self& _z)
5	template <class _Tp2> explicit complex(const complex<_Tp2>& _z)
6	_Self& operator=(const _Self& _z)
7	template <class _Tp2> _Self& operator=(const complex<_Tp2>& _z)
8	value_type real()
9	value_type imag()
10	_Self& operator= (const value_type& _x)
11	_Self& operator+= (const value_type& _x)
12	_Self& operator-= (const value_type& _x)
13	_Self& operator*= (const value_type& _x)
14	_Self& operator/= (const value_type& _x)
15	static void _STLP_CALL _div(const value_type& _z1_r, const value_type& _z1_i, const value_type& _z2_r, const value_type& _z2_i, value_type& _res_r, value_type& _res_i);
16	static void _STLP_CALL _div(const value_type& _z1_r, const value_type& _z2_r, const value_type& _z2_i, vaue_type& _res_r, es_i);
17	template <class _Tp2> _Self& operator+= (const complex<_Tp2>& _z)
18	template <class _Tp2> _Self& operator-= (const complex<_Tp2>& _z)
19	template <class _Tp2> _Self& operator*= (const complex<_Tp2>& _z)
20	template <class _Tp2> _Self& operator/= (const complex<_Tp2>& _z)
21	_Self& operator+= (const _Self& _z)
22	_Self& operator-= (const _Self& _z)
23	_Self& operator*= (const _Self& _z)
24	_Self& operator/= (const _Self& _z)
25	complex(value_type _x = 0.0f, value_type _y = 0.0f)
26	complex(const complex<float>& _z)
27	inline explicit complex(const complex<double>& _z);
28	inline explicit complex(const complex<long double>& _z)
29	value_type real()
30	value_type imag()
31	_Self& operator= (const value_type& _x)
32	_Self& operator+= (const value_type& _x)
33	_Self& operator-= (const value_type& _x)
34	_Self& operator*= (const value_type& _x)
35	_Self& operator/= (const value_type& _x)
36	static void _STLP_CALL _div(const value_type& _z1_r, const value_type& _z1_i, const value_type& _z2_r, const value_type& _z2_i, value_type& _res_r, value_type& _res_i);
37	static void _STLP_CALL _div(const value_type& _z1_r, const value_type& _z2_r, const value_type& _z2_i, vaue_type& _res_r, es_i);
38	template <class _Tp2> complex<float>& operator=(const complex<_Tp2>& _z)
39	template <class _Tp2> complex<float>& operator+= (const complex<_Tp2>& _z)
40	template <class _Tp2> complex<float>& operator-= (const complex<_Tp2>& _z)
41	template <class _Tp2> complex<float>& operator*= (const complex<_Tp2>& _z)
42	template <class _Tp2> complex<float>& operator/= (const complex<_Tp2>& _z)
43	_Self& operator=(const _Self& _z)
44	_Self& operator+= (const _Self& _z)
45	_Self& operator-= (const _Self& _z)
46	_Self& operator*= (const _Self& _z)

47	_Self& operator/= (const _Self& _z)
48	complex(value_type _x = 0.0, value_type _y = 0.0)
49	complex(const complex<double>& _z)
50	inline complex(const complex<float>& _z)
51	inline explicit complex(const complex<long double>& _z)
52	value_type real()
53	value_type imag()
54	_Self& operator= (const value_type& _x)
55	_Self& operator+= (const value_type& _x)
56	_Self& operator-= (const value_type& _x)
57	_Self& operator*= (const value_type& _x)
58	_Self& operator/= (const value_type& _x)
59	static void _STLP_CALL _div(const value_type& _z1_r, const value_type& _z1_i, const value_type& _z2_r, const value_type& _z2_i, value_type& _res_r, value_type& _res_i);
60	static void _STLP_CALL _div(const value_type& _z1_r, const value_type& _z2_r, const value_type& _z2_i, vaue_type& _res_r, es_i);
61	template <class _Tp2> complex<double>& operator=(const complex<_Tp2>& _z)
62	template <class _Tp2> complex<double>& operator+= (const complex<_Tp2>& _z)
63	template <class _Tp2> complex<double>& operator-= (const complex<_Tp2>& _z)
64	template <class _Tp2> complex<double>& operator*= (const complex<_Tp2>& _z)
65	template <class _Tp2> complex<double>& operator/= (const complex<_Tp2>& _z)
66	_Self& operator=(const _Self& _z)
67	_Self& operator+= (const _Self& _z)
68	_Self& operator-= (const _Self& _z)
69	_Self& operator*= (const _Self& _z)
70	_Self& operator/= (const _Self& _z)
71	complex(value_type _x = 0.0, value_type _y = 0.0)
72	complex(const complex<double>& _z)
73	inline complex(const complex<float>& _z)
74	inline explicit complex(const complex<long double>& _z)
75	value_type real()
76	value_type imag()
77	_Self& operator= (const value_type& _x)
78	_Self& operator+= (const value_type& _x)
79	_Self& operator-= (const value_type& _x)
80	_Self& operator*= (const value_type& _x)
81	_Self& operator/= (const value_type& _x)
82	static void _STLP_CALL _div(const value_type& _z1_r, const value_type& _z1_i, const value_type& _z2_r, const value_type& _z2_i, value_type& _res_r, value_type& _res_i);
83	static void _STLP_CALL _div(const value_type& _z1_r, const value_type& _z2_r, const value_type& _z2_i, vaue_type& _res_r, es_i);
84	template <class _Tp2> complex<double>& operator=(const complex<_Tp2>& _z)
85	template <class _Tp2> complex<double>& operator+= (const complex<_Tp2>& _z)
86	template <class _Tp2> complex<double>& operator-= (const complex<_Tp2>& _z)
87	template <class _Tp2> complex<double>& operator*= (const complex<_Tp2>& _z)
88	template <class _Tp2> complex<double>& operator/= (const complex<_Tp2>& _z)
89	_Self& operator=(const _Self& _z)
90	_Self& operator+= (const _Self& _z)
91	_Self& operator-= (const _Self& _z)
92	_Self& operator*= (const _Self& _z)
93	_Self& operator/= (const _Self& _z)
94	inline complex<float>::complex(const complex<double>& _z)
95	inline complex<double>::complex(const complex<float>& _z)
96	inline complex<float>::complex(const complex<long double>& _z)
97	inline complex<double>::complex(const complex<long double>& _z)

98	inline complex<long double>::complex(const complex<float>& _z)
99	inline complex<long double>::complex(const complex<double>& _z)
100	template <class _Tp> inline complex<_Tp> _STLP_CALL operator+(const complex<_Tp>& _z)
101	template <class _Tp> inline complex<_Tp> _STLP_CALL　operator−(const complex<_Tp>& _z)
102	template <class _Tp> inline complex<_Tp> _STLP_CALL operator+(const _Tp& _x, const complex<_Tp>& _z)
103	template <class _Tp> inline complex<_Tp> _STLP_CALL operator+(const complex<_Tp>& _z, const _Tp& _x)
104	template <class _Tp> inline complex<_Tp> _STLP_CALL operator−(const _Tp& _x, const complex<_Tp>& _z)
105	template <class _Tp> inline complex<_Tp> _STLP_CALL operator−(const complex<_Tp>& _z, const _Tp& _x)
106	template <class _Tp> inline complex<_Tp> _STLP_CALL operator*(const _Tp& _x, const complex<_Tp>& _z)
107	template <class _Tp> inline complex<_Tp> _STLP_CALL operator*(const complex<_Tp>& _z, const _Tp& _x)
108	template <class _Tp> inline complex<_Tp> _STLP_CALL operator/(const _Tp& _x, const complex<_Tp>& _z)
109	template <class _Tp> inline complex<_Tp> _STLP_CALL operator/(const complex<_Tp>& _z, const _Tp& _x)
110	template <class _Tp> inline complex<_Tp> _STLP_CALL operator+(const complex<_Tp>& _z1, const complex<_Tp>& _z2)
111	template <class _Tp> inline complex<_Tp> _STLP_CALL operator−(const complex<_Tp>& _z1, const complex<_Tp>& _z2)
112	template <class _Tp> inline complex<_Tp> _STLP_CALL operator*(const complex<_Tp>& _z1, const complex<_Tp>& _z2)
113	template <class _Tp> inline complex<_Tp> _STLP_CALL operator/(const complex<_Tp>& _z1, const complex<_Tp>& _z2)
114	template <class _Tp> inline bool _STLP_CALL operator==(const complex<_Tp>& _z1, const complex<_Tp>& _z2)
115	template <class _Tp> inline bool _STLP_CALL operator==(const complex<_Tp>& _z, const _Tp& _x)
116	template <class _Tp> inline bool _STLP_CALL operator==(const _Tp& _x, const complex<_Tp>& _z)
117	template <class _Tp> inline bool _STLP_CALL operator!=(const complex<_Tp>& _z1, const complex<_Tp>& _z2)
118	template <class _Tp> inline bool _STLP_CALL operator!=(const complex<_Tp>& _z, const _Tp& _x)
119	template <class _Tp> inline bool _STLP_CALL operator!=(const _Tp& _x, const complex<_Tp>& _z)
120	template <class _Tp> inline _Tp _STLP_CALL real(const complex<_Tp>& _z)
121	template <class _Tp> inline _Tp _STLP_CALL imag(const complex<_Tp>& _z)
122	template <class _Tp> _Tp _STLP_CALL abs(const complex<_Tp>& _z)
123	template <class _Tp> _Tp _STLP_CALL arg(const complex<_Tp>& _z)
124	template <class _Tp> inline _Tp _STLP_CALL norm(const complex<_Tp>& _z)
125	template <class _Tp> inline complex<_Tp> _STLP_CALL conj(const complex<_Tp>& _z)
126	template <class _Tp> complex<_Tp> _STLP_CALL polar(const _Tp& _rho)
127	template <class _Tp> complex<_Tp> _STLP_CALL polar(const _Tp& _rho, const _Tp& _phi)
128	float _STLP_CALL abs(const complex<float>&)
129	double _STLP_CALL abs(const complex<double>&)

130	_STLP_DECLSPEC long double _STLP_CALL abs(const complex<long double>&)
131	float _STLP_CALL arg(const complex<float>&)
132	double _STLP_CALL arg(const complex<double>&)
133	_STLP_DECLSPEC long double _STLP_CALL arg(const complex<long double>&)
134	complex<float> _STLP_CALL polar(const float& _rho, const float& _phi)
135	complex<double> _STLP_CALL polar(const double& _rho, const double& _phi)
136	_STLP_DECLSPEC complex<long double> _STLP_CALL polar(const long double&, const long double&)
137	template <class _Tp> _Tp _STLP_CALL abs(const complex<_Tp>& _z)
138	template <class _Tp> _Tp _STLP_CALL arg(const complex<_Tp>& _z)
139	template <class _Tp> complex<_Tp> _STLP_CALL polar(const _Tp& _rho, const _Tp& _phi)
140	_STLP_DECLSPEC complex<float> _STLP_CALL sqrt(const complex<float>&)
141	_STLP_DECLSPEC complex<float> _STLP_CALL exp(const complex<float>&)
142	_STLP_DECLSPEC complex<float> _STLP_CALL log(const complex<float>&)
143	_STLP_DECLSPEC complex<float> _STLP_CALL log10(const complex<float>&)
144	_STLP_DECLSPEC complex<float> _STLP_CALL pow(const complex<float>&, int)
145	_STLP_DECLSPEC complex<float> _STLP_CALL pow(const complex<float>&, const float&)
146	_STLP_DECLSPEC complex<float> _STLP_CALL pow(const float&, const complex<float>&)
147	_STLP_DECLSPEC complex<float> _STLP_CALL pow(const complex<float>&, const complex<float>&)
148	_STLP_DECLSPEC complex<float> _STLP_CALL sin(const complex<float>&)
149	_STLP_DECLSPEC complex<float> _STLP_CALL cos(const complex<float>&)
150	_STLP_DECLSPEC complex<float> _STLP_CALL tan(const complex<float>&)
151	_STLP_DECLSPEC complex<float> _STLP_CALL sinh(const complex<float>&)
152	_STLP_DECLSPEC complex<float> _STLP_CALL cosh(const complex<float>&)
153	_STLP_DECLSPEC complex<float> _STLP_CALL tanh(const complex<float>&)
154	_STLP_DECLSPEC complex<double> _STLP_CALL sqrt(const complex<double>&)
155	_STLP_DECLSPEC complex<double> _STLP_CALL exp(const complex<double>&)
156	_STLP_DECLSPEC complex<double> _STLP_CALL log(const complex<double>&)
157	_STLP_DECLSPEC complex<double> _STLP_CALL log10(const complex<double>&)
158	_STLP_DECLSPEC complex<double> _STLP_CALL pow(const complex<double>&, int)
159	_STLP_DECLSPEC complex<double> _STLP_CALL pow(const complex<double>&, const double&)
160	_STLP_DECLSPEC complex<double> _STLP_CALL pow(const double&, const complex<double>&)
161	_STLP_DECLSPEC complex<double> _STLP_CALL pow(const complex<double>&, const complex<double>&)
162	_STLP_DECLSPEC complex<double> _STLP_CALL sin(const complex<double>&)
163	_STLP_DECLSPEC complex<double> _STLP_CALL cos(const complex<double>&);
164	_STLP_DECLSPEC complex<double> _STLP_CALL tan(const complex<double>&)
165	_STLP_DECLSPEC complex<double> _STLP_CALL sinh(const complex<double>&);
166	_STLP_DECLSPEC complex<double> _STLP_CALL cosh(const complex<double>&);
167	_STLP_DECLSPEC complex<double> _STLP_CALL tanh(const complex<double>&);
168	_STLP_DECLSPEC complex<long double> _STLP_CALL sqrt(const complex<long double>&)
169	_STLP_DECLSPEC complex<long double> _STLP_CALL exp(const complex<long double>&)
170	_STLP_DECLSPEC complex<long double> _STLP_CALL log(const complex<long double>&)
171	_STLP_DECLSPEC complex<long double> _STLP_CALL log10(const complex<long double>&)
172	_STLP_DECLSPEC complex<long double> _STLP_CALL pow(const complex<long double>&, int)
173	_STLP_DECLSPEC complex<long double> _STLP_CALL pow(const complex<long double>&, const long double&)
174	_STLP_DECLSPEC complex<long double> _STLP_CALL pow(const long double&, const complex<long double>&)
175	_STLP_DECLSPEC complex<long double> _STLP_CALL pow(const complex<long double>&, const complex<long double>&)
176	_STLP_DECLSPEC complex<long double> _STLP_CALL sin(const complex<long double>&)
177	_STLP_DECLSPEC complex<long double> _STLP_CALL cos(const complex<long double>&);
178	_STLP_DECLSPEC complex<long double> _STLP_CALL tan(const complex<long double>&)
179	_STLP_DECLSPEC complex<long double> _STLP_CALL sinh(const complex<long double>&);
180	_STLP_DECLSPEC complex<long double> _STLP_CALL cosh(const complex<long double>&);
181	_STLP_DECLSPEC complex<long double> _STLP_CALL tanh(const complex<long double>&);

5. <deque>

<deque>でサポートしているメソッドは以下になります。

deque クラスのメソッド一覧	
1	iterator begin()
2	const_iterator begin()
3	iterator end()
4	const_iterator end()
5	reverse_iterator rbegin()
6	const_reverse_iterator rbegin()
7	reverse_iterator rend()
8	const_reverse_iterator rend()
9	reference operator[](size_type _n)
10	const_reference operator[](size_type _n)
11	void _M_range_check(size_type _n)
12	reference at(size_type _n)
13	const_reference at(size_type _n)
14	reference front()
15	const_reference front()
16	reference back()
17	const_reference back()
18	size_type size()
19	size_type max_size()
20	bool empty()
21	allocator_type get_allocator()
22	deque()
23	deque(const allocator_type& _a)
24	deque(const _Self& _x)
25	explicit deque(size_type _n)
26	deque(_InputIterator _first, _InputIterator _last, const allocator_type& _a = allocator_type())
27	deque(__move_source<_Self> src)
28	~deque()
29	void swap(_Self& _x)
30	void _M_fill_assign(size_type _n, const _Tp& _val)
31	void assign(size_type _n, const _Tp& _val)
32	void assign(_InputIterator _first, _InputIterator _last)
34	void push_back(const value_type& _t = _Tp())
35	void push_front(const value_type& _t = _Tp())
36	void pop_back()
37	void pop_front()
38	iterator insert(iterator _pos, const value_type& _x = _Tp())
39	void insert(iterator _pos, size_type _n, const value_type& _x)
40	void insert(iterator _pos, _InputIterator _first, _InputIterator _last)
41	void clear()
42	void resize(size_type _new_size,const value_type& _x = _Tp())
43	iterator erase(iterator _pos)
44	iterator erase(iterator _first, iterator _last)
45	_Self &　operator= (const _Self &_x)
46	_Self &　operator!= (const _Self &_x)
47	_Self &　operator<(const _Self &_x)
48	_Self &　operator<= (const _Self &_x)
49	_Self &　operator== (const _Self &_x)
50	_Self &　operator>(const _Self &_x)
51	_Self &　operator>= (const _Self &_x)

6. <exception>

<exception>でサポートしているメソッドは以下になります。

exception クラスのメソッド一覧	
exception サブクラスのメソッド一覧	
1	exception()
2	virtual ~exception()
3	const char* what()
bad_alloc サブクラスのメソッド一覧	
4	bad_alloc ()
5	bad_alloc(const bad_alloc&)
6	bad_alloc& operator=(const bad_alloc&)
7	~bad_alloc ()
8	what()
bad_exception サブクラスのメソッド一覧	
9	bad_exception()
10	~bad_exception()
11	what()
bad_cast サブクラスのメソッド一覧	
12	bad_cast()
13	bad_cast(const bad_cast&)
14	bad_cast& operator=(const bad_cast&)
15	~bad_cast()
16	const char* what()
bad_typeid サブクラスのメソッド一覧	
17	bad_typeid()
18	bad_typeid(const bad_typeid&)
19	bad_typeid& operator=(const bad_typeid&)
20	~bad_typeid()
21	const char* what()

7. <functional>

<functional>でサポートしているメソッドは以下になります。

functional クラスのメソッド一覧	
binary_function 基底クラスのメソッド一覧	
1	template <class _Tp> struct not_equal_to : public binary_function<_Tp, _Tp, bool> { bool operator()(const _Tp& _x, const _Tp& _y) const
2	template <class _Tp> struct greater : public binary_function<_Tp, _Tp, bool> { bool operator()(const _Tp& _x, const _Tp& _y)
3	emplate <class _Tp> struct greater_equal : public binary_function<_Tp, _Tp, bool> { bool operator()(const _Tp& _x, const _Tp& _y)
4	template <class _Tp> struct less_equal : public binary_function<_Tp, _Tp, bool> { bool operator()(const _Tp& _x, const _Tp& _y)
5	template <class _Tp> struct divides : public binary_function<_Tp, _Tp, _Tp> { _Tp operator()(const _Tp& _x, const _Tp& _y)
6	template <class _Tp> struct modulus : public binary_function<_Tp, _Tp, _Tp> { _Tp operator()(const _Tp& _x, const _Tp& _y)
7	template <class _Tp> struct logical_and : public binary_function<_Tp, _Tp, bool> { bool operator()(const _Tp& _x, const _Tp& _y)
8	template <class _Tp> struct logical_or : public binary_function<_Tp, _Tp,bool> { bool operator()(const _Tp& _x, const _Tp& _y) const
unary_function サブクラスのメソッド一覧	
9	template <class _Tp> struct negate : public unary_function<_Tp, _Tp> { _Tp operator()(const _Tp& _x)
10	template <class _Tp> struct logical_not : public unary_function<_Tp, bool> { bool operator()(const _Tp& _x)
class unary_negate サブクラスのメソッド一覧	
11	explicit unary_negate(const _Predicate& _x)
12	bool operator()(_ArgParamType _x)
13	template <class _Predicate> inline unary_negate<_Predicate> not1(const _Predicate& _pred)
binary_negate サブクラスのメソッド一覧	
14	explicit binary_negate(const _Predicate& _x)
15	bool operator()(_FstArgParamType _x, _SndArgParamType _y)
16	binary_negate<_Predicate> not2(const _Predicate& _pred)
binder1st サブクラスのメソッド一覧	
17	binder1st(const _Operation& _x, _ValueParamType _y)
18	result_type operator()(_ConstArgParamType _x)
19	result_type operator()(_ArgParamType _x)
class binder2nd サブクラスのメソッド一覧	
20	binder2nd(const _Operation& _x, _ValueParamType _y)
21	result_type operator()(_ConstArgParamType _x)
22	result_type operator()(_ArgParamType _x)
class unary_compose サブクラスのメソッド一覧	
23	unary_compose(const _Operation1& _x, const _Operation2& _y)
24	result_type operator()(_ArgParamType _x)
class binary_compose サブクラスのメソッド一覧	
25	binary_compose(const _Operation1& _x, const _Operation2& _y, const _Operation3& _z)
26	result_type operator()(_ArgParamType _x)

27	template <class _Operation1, class _Operation2, class _Operation3> inline binary_compose<_Operation1, _Operation2, _Operation3> compose2(const _Operation1& _fn1, const _Operation2& _fn2, const _Operation3& _fn3)
class constant_void_fun サブクラスのメソッド一覧	
28	_Constant_void_fun(const result_type& __v)
29	const result_type& operator()
30	template <class _Result> struct constant_void_fun : public _STLP_PRIV _Constant_void_fun<_Result> { constant_void_fun(const _Result& __v)
31	template <class _Result, _STLP_DFL_TMPL_PARAM(_Argument , _Result) > struct constant_unary_fun : public _STLP_PRIV _Constant_unary_fun<_Result, _Argument> { constant_unary_fun(const _Result& __v)
32	template <class _Result, _STLP_DFL_TMPL_PARAM(_Arg1 , _Result), _STLP_DFL_TMPL_PARAM(_Arg2 , _Arg1) > struct constant_binary_fun : public _STLP_PRIV _Constant_binary_fun<_Result, _Arg1, _Arg2> { constant_binary_fun(const _Result& __v)
33	template <class _Result> inline constant_void_fun<_Result> constant0(const _Result& __val)
34	template <class _Result> inline constant_unary_fun<_Result,_Result> constant1(const _Result& __val)
35	template <class _Result> inline constant_binary_fun<_Result,_Result,_Result> constant2(const _Result& __val)
class subtractive_rng サブクラスのメソッド一覧	
36	_STLP_UINT32_T operator()(_STLP_UINT32_T __limit)
37	void _M_initialize(_STLP_UINT32_T __seed)
38	subtractive_rng(unsigned int __seed)

8. <iterator>

<iterator>でサポートしているメソッドは以下になります。

iterator クラスのメソッド一覧	
class reverse_iterator サブクラスのメソッド一覧	
1	reverse_iterator()
2	explicit reverse_iterator(iterator_type _x)
3	reverse_iterator(const _Self& _x)
4	template <class _Iter> reverse_iterator(const reverse_iterator<_Iter>& _x)
5	_Self& operator = (const _Self& _x)
6	_Self& operator = (const reverse_iterator<_Iter>& _x)
7	iterator_type base()
8	reference operator*()
9	_Self& operator++()
10	_Self operator++(int)
11	_Self& operator--()
12	_Self operator--(int)
13	_Self operator+(difference_type _n)
14	_Self& operator+=(difference_type _n)
15	_Self operator-(difference_type _n)
16	_Self& operator-=(difference_type _n)
17	reference operator[](difference_type _n)
18	template <class _Iterator> inline bool _STLP_CALL operator==(const reverse_iterator<_Iterator>& _x, const reverse_iterator<_Iterator>& _y)
19	template <class _Iterator> inline bool _STLP_CALL operator<(const reverse_iterator<_Iterator>& _x, const reverse_iterator<_Iterator>& _y)
20	template <class _Iterator> inline bool _STLP_CALL operator!=(const reverse_iterator<_Iterator>& _x, const reverse_iterator<_Iterator>& _y)
21	template <class _Iterator> inline bool _STLP_CALL operator>(const reverse_iterator<_Iterator>& _x, const reverse_iterator<_Iterator>& _y)
22	template <class _Iterator> inline bool _STLP_CALL operator<=(const reverse_iterator<_Iterator>& _x, const reverse_iterator<_Iterator>& _y)
23	template <class _Iterator> inline bool _STLP_CALL operator>=(const reverse_iterator<_Iterator>& _x, const reverse_iterator<_Iterator>& _y)
class back_insert_iterator サブクラスのメソッド一覧	
24	explicit back_insert_iterator(_Container& _x)
25	_Self& operator=(const _Self& _other)
26	_Self& operator=(const typename _Container::value_type& _val)
27	_Self& operator*()
28	_Self& operator++()
29	_Self operator++(int)
front_insert_iterator サブクラスのメソッド一覧	
30	explicit front_insert_iterator(_Container& _x)
31	_Self& operator=(const _Self& _other)
32	_Self& operator=(const typename _Container::value_type& _val)
33	_Self& operator*()
34	_Self& operator++()
35	_Self operator++(int)
insert_iterator サブクラスのメソッド一覧	
36	insert_iterator(_Container& _x, typename _Container::iterator _i)
37	_Self& operator=(_Self const& _other)
38	_Self& operator=(const typename _Container::value_type& _val)
39	_Self& operator*()
40	_Self& operator++()
41	_Self operator++(int)

9. <limits>

<limits>でサポートしているメソッドは以下になります。

limits クラスのメソッド一覧	
1	static _number (_STLP_CALL min)()
2	static _number (_STLP_CALL max)()
3	_STLP_STATIC_CONSTANT(int, digits = 0)
4	_STLP_STATIC_CONSTANT(int, digits10 = 0)
5	_STLP_STATIC_CONSTANT(int, radix = 0)
6	_STLP_STATIC_CONSTANT(int, min_exponent = 0)
7	_STLP_STATIC_CONSTANT(int, min_exponent10 = 0)
8	_STLP_STATIC_CONSTANT(int, max_exponent = 0)
9	_STLP_STATIC_CONSTANT(int, max_exponent10 = 0)
10	_STLP_STATIC_CONSTANT(float_denorm_style, has_denorm = denorm_absent)
11	_STLP_STATIC_CONSTANT(float_round_style, round_style = round_toward_zero)
12	_STLP_STATIC_CONSTANT(bool, is_specialized = false)
13	_STLP_STATIC_CONSTANT(bool, is_signed = false)
14	_STLP_STATIC_CONSTANT(bool, is_integer = false)
15	_STLP_STATIC_CONSTANT(bool, is_exact = false)
16	_STLP_STATIC_CONSTANT(bool, has_infinity = false)
17	_STLP_STATIC_CONSTANT(bool, has_quiet_NaN = false)
18	_STLP_STATIC_CONSTANT(bool, has_signaling_NaN = false)
19	_STLP_STATIC_CONSTANT(bool, has_denorm_loss = false)
20	_STLP_STATIC_CONSTANT(bool, is_iec559 = false)
21	_STLP_STATIC_CONSTANT(bool, is_bounded = false)
22	_STLP_STATIC_CONSTANT(bool, is_modulo = false)
23	_STLP_STATIC_CONSTANT(bool, traps = false)
24	_STLP_STATIC_CONSTANT(bool, tinyness_before = false)
25	_number _STLP_CALL epsilon()
26	_number _STLP_CALL round_error()
27	_number _STLP_CALL infinity()
28	_number _STLP_CALL quiet_NaN()
29	_number _STLP_CALL signaling_NaN()
30	_number _STLP_CALL denorm_min()

10. <list>

<list>でサポートしているメソッドは以下になります。

list クラスのメソッド一覧	
1	list(size_type _n, const_reference _val = _STLP_DEFAULT_CONSTRUCTED(value_type), const allocator_type& _a = allocator_type())
2	list(_InputIterator _first, _InputIterator _last, const allocator_type& _a _STLP_ALLOCATOR_TYPE_DFL)
3	list(const allocator_type& _a = allocator_type())
4	list(const _Self& _x)
5	list(_move_source<_Self> src)
6	void clear()
7	bool empty()
8	iterator begin()
9	const_iterator begin()
10	iterator end()
11	const_iterator end()
12	reverse_iterator rbegin()
13	const_reverse_iterator rbegin()
14	reverse_iterator rend()
15	const_reverse_iterator rend()
16	size_type size()
17	size_type max_size()
18	reference front()
19	const_reference front()
20	reference back()
21	const_reference back()
22	void swap(_Self& _x)
23	iterator insert(iterator _pos, const_reference _x = value_type())
24	void insert(iterator _pos, _InputIterator _first, _InputIterator _last)
25	void insert(iterator _pos, size_type _n, const_reference _x)
26	void push_front(const_reference _x)
27	void push_back (const_reference _x)
28	iterator erase(iterator _pos)
29	iterator erase(iterator _first, iterator _last)
30	void resize(size_type _new_size, const_reference _x = value_type())
31	void pop_front()
32	void pop_back()
33	void assign(size_type _n, const_reference _val)
34	void assign(_InputIterator _first, _InputIterator _last)
35	void splice(iterator _pos, _Self& _x)
36	void splice(iterator _pos, _Self& _x, iterator _i)
37	void splice(iterator _pos, _Self& _x, iterator _first, iterator _last)
38	void remove(const_reference _val)
39	void unique()
40	void merge(_Self& _x)
41	void reverse()
42	void sort()
43	_Self& operator=
44	bool _STLP_CALL operator==
45	bool _STLP_CALL operator!=
46	bool _STLP_CALL operator<
47	bool _STLP_CALL operator<=
48	bool _STLP_CALL operator>
49	bool _STLP_CALL operator>=
50	allocator_type get_allocator()
51	void _M_fill_assign(size_type _n, const_reference _val)

11. <map>

<map>でサポートしているメソッドは以下になります。

map クラスのメソッド一覧	
1	map()
2	map(const _Compare& __comp, const allocator_type& __a = allocator_type())
3	map(_InputIterator __first, _InputIterator __last)
4	map(_InputIterator __first, _InputIterator __last, const _Compare& __comp, const allocator_type& __a _STLP_ALLOCATOR_TYPE_DFL)
5	map(const _Self& __x)
6	map(__move_source<_Self> src)
7	key_compare key_comp()
8	value_compare value_comp()
9	allocator_type get_allocator()
10	iterator begin()
11	const_iterator begin()
12	iterator end()
13	const_iterator end()
14	reverse_iterator rbegin()
15	const_reverse_iterator rbegin()
16	reverse_iterator rend()
17	const_reverse_iterator rend()
18	bool empty()
19	size_type size()
20	size_type max_size()
21	void swap(_Self& __x)
22	pair<iterator,bool> insert(const value_type& __x)
23	iterator insert(iterator __pos, const value_type& __x)
24	void insert(_InputIterator __first, _InputIterator __last)
25	void erase(iterator __pos)
26	size_type erase(const key_type& __x)
27	void erase(iterator __first, iterator __last)
28	void clear()
29	iterator find(const _KT& __x)
30	const_iterator find(const _KT& __x)
31	size_type count(const _KT& __x)
32	iterator lower_bound(const _KT& __x)
33	const_iterator lower_bound(const _KT& __x)
34	iterator upper_bound(const _KT& __x)
35	const_iterator upper_bound(const _KT& __x)
36	pair<iterator,iterator> equal_range(const _KT& __x)
37	pair<const_iterator,const_iterator> equal_range(const _KT& __x)
38	_Self& operator= (const _Self& __x)
39	_Tp& operator[] (const _KT& __k)
40	bool _STLP_CALL operator!= (const map<_Key,_Tp,_Compare,_Alloc>& __x, const map<_Key,_Tp,_Compare,_Alloc>&__y)
41	bool _STLP_CALL operator< (const map<_Key,_Tp,_Compare,_Alloc>& __x, const map<_Key,_Tp,_Compare,_Alloc>&__y)
42	bool _STLP_CALL operator<= (const map<_Key,_Tp,_Compare,_Alloc>& __x, const map<_Key,_Tp,_Compare,_Alloc>&__y)
43	bool _STLP_CALL operator== (const map<_Key,_Tp,_Compare,_Alloc>& __x, const map<_Key,_Tp,_Compare,_Alloc>&__y)
44	bool _STLP_CALL operator> (const map<_Key,_Tp,_Compare,_Alloc>& __x, const map<_Key,_Tp,_Compare,_Alloc>&__y)
45	bool _STLP_CALL operator>= (const map<_Key,_Tp,_Compare,_Alloc>& __x, const map<_Key,_Tp,_Compare,_Alloc>&__y)

multimap クラスのメソッド一覧	
1	multimap()
2	multimap(const _Compare& __comp, const allocator_type& __a = allocator_type())
3	multimap(_InputIterator __first, _InputIterator __last)
4	multimap(_InputIterator __first, _InputIterator __last, const _Compare& __comp, const allocator_type& __a _STLP_ALLOCATOR_TYPE_DFL)
5	multimap(const _Self& __x)
6	multimap(__move_source<_Self> src)
7	key_compare key_comp()
8	value_compare value_comp()
9	allocator_type get_allocator()
10	iterator begin()
11	const_iterator begin()
12	iterator end()
13	const_iterator end()
14	reverse_iterator rbegin()
15	const_reverse_iterator rbegin()
16	reverse_iterator rend()
17	const_reverse_iterator rend()
18	bool empty()
19	size_type size()
20	size_type max_size()
21	void swap(_Self& __x)
22	iterator insert(const value_type& __x)
23	iterator insert(iterator __pos, const value_type& __x)
24	void insert(_InputIterator __first, _InputIterator __last)
25	void erase(iterator __pos)
26	size_type erase(const key_type& __x)
27	void erase(iterator __first, iterator __last)
28	void clear()
29	iterator find(const _KT& __x)
30	const_iterator find(const _KT& __x)
31	size_type count(const _KT& __x)
32	iterator lower_bound(const _KT& __x)
33	const_iterator lower_bound(const _KT& __x)
34	iterator upper_bound(const _KT& __x)
35	const_iterator upper_bound(const _KT& __x)
36	pair<iterator,iterator> equal_range(const _KT& __x)
37	pair<const_iterator,const_iterator> equal_range(const _KT& __x)
38	_Self& operator= (const _Self& __x)
39	bool _STLP_CALL operator!= (const multimap<_Key,_Tp,_Compare,_Alloc>& __x, const multimap<_Key,_Tp,_Compare,_Alloc>&__y)
40	bool _STLP_CALL operator< (const multimap<_Key,_Tp,_Compare,_Alloc>& __x, const multimap<_Key,_Tp,_Compare,_Alloc>&__y)
41	bool _STLP_CALL operator<= (const multimap<_Key,_Tp,_Compare,_Alloc>& __x, const multimap<_Key,_Tp,_Compare,_Alloc>&__y)
42	bool _STLP_CALL operator== (const multimap<_Key,_Tp,_Compare,_Alloc>& __x, const multimap<_Key,_Tp,_Compare,_Alloc>&__y)
43	bool _STLP_CALL operator> (const multimap<_Key,_Tp,_Compare,_Alloc>& __x, const multimap<_Key,_Tp,_Compare,_Alloc>&__y)
44	bool _STLP_CALL operator>= (const multimap<_Key,_Tp,_Compare,_Alloc>& __x, const multimap<_Key,_Tp,_Compare,_Alloc>&__y)

12. <memory>

<memory>でサポートしているメソッドは以下になります。

allocator クラスのメソッド一覧	
1	allocator()
2	allocator(const allocator<_Tp1>&)
3	allocator(__move_source<allocator<_Tp> > src)
4	~allocator()
5	pointer address(reference __x)
6	const_pointer address(const_reference __x)
7	_Tp* allocate(size_type __n, const void* = 0)
8	void deallocate(pointer __p, size_type __n)
9	void deallocate(pointer __p) const
10	size_type max_size() const
11	void construct(pointer __p, const_reference __val)
12	void destroy(pointer __p)

auto_ptr クラスのメソッド一覧	
1	_Tp* release()
2	void reset(_Tp* __px = 0)
3	_Tp* get()
4	auto_ptr(_Tp* __px = 0)
5	auto_ptr(_Self& __r)
6	auto_ptr(auto_ptr_ref<_Tp> __r)
7	operator auto_ptr_ref<_Tp1>()
8	_Tp* operator->()
9	_Tp* operator*()
10	_Self& operator= (_Self& __r)
11	_Self& operator= (auto_ptr_ref<_Tp> __r)
12	operator auto_ptr<_Tp1>()

auto_ptr_ref クラスのメソッド一覧	
1	auto_ptr_ref(__ptr_base& __r, _Tp* __p)
2	_Tp* release()

raw_storage_iterator クラスのメソッド一覧	
1	raw_storage_iterator(_ForwardIterator __x)
2	raw_storage_iterator<_ForwardIterator, _Tp>& operator*()
3	raw_storage_iterator<_ForwardIterator, _Tp>& operator=(const _Tp& __element)
4	raw_storage_iterator<_ForwardIterator, _Tp>& operator++()
5	raw_storage_iterator<_ForwardIterator, _Tp> operator++(int)

13. <numeric>

<numeric>でサポートしているメソッドは以下になります。

numeric クラスのメソッド一覧	
1	_Tp accumulate(_InputIterator _first, _InputIterator _last, _Tp _Init)
2	_Tp accumulate(_InputIterator _first, _InputIterator _last, _Tp _Init, _BinaryOperation _binary_op)
3	_Tp inner_product(_InputIterator1 _first1, _InputIterator1 _last1, _InputIterator2 _first2, _Tp _Init)
4	_Tp inner_product(_InputIterator1 _first1, _InputIterator1 _last1, _InputIterator2 _first2, _Tp _Init, _BinaryOperation1 _binary_op1, _BinaryOperation2 _binary_op2)
5	_OutputIterator partial_sum(_InputIterator _first, _InputIterator _last, _OutputIterator _result)
6	_OutputIterator partial_sum(_InputIterator _first, _InputIterator _last, _OutputIterator _result, _BinaryOperation _binary_op)
7	_OutputIterator adjacent_difference(_InputIterator _first, _InputIterator _last, _OutputIterator _result)
8	_OutputIterator adjacent_difference(_InputIterator _first, _InputIterator _last, _OutputIterator _result, _BinaryOperation _binary_op)
9	_Tp power(_Tp _x, _Integer _n, _MonoidOperation _opr)
10	_Tp power(_Tp _x, _Integer _n)
11	void iota(_ForwardIterator _first, _ForwardIterator _last, _Tp _val)

14. <queue>

<queue>でサポートしているメソッドは以下になります。

queue クラスのメソッド一覧	
1	queue()
2	queue(const _Sequence& _c)
3	queue(_move_source<_Self> src)
4	bool empty()
5	size_type size()
6	reference front()
7	const_reference front()
8	reference back()
9	const_reference back()
10	void push(const value_type& _x)
11	void pop()
12	const _Sequence& _Get_s()
13	bool _STLP_CALL operator==(const queue<_Tp>& _x, const queue<_Tp>& _y)
14	bool _STLP_CALL operator<(const queue<_Tp>& _x, const queue<_Tp>& _y)
15	bool _STLP_CALL operator>(const queue<_Tp>& _x, const queue<_Tp>& _y)
16	bool _STLP_CALL operator<=(const queue<_Tp>& _x, const queue<_Tp>& _y)
17	bool _STLP_CALL operator>=(const queue<_Tp>& _x, const queue<_Tp>& _y)
18	bool _STLP_CALL operator!=(const queue<_Tp>& _x, const queue<_Tp>& _y)

priority_queue クラスのメソッド一覧	
1	priority_queue()
2	priority_queue(const _Compare& _x)
3	priority_queue(const _Compare& _x, const _Sequence& _s)
4	priority_queue(_move_source<_Self> src)
5	priority_queue(_InputIterator _first, _InputIterator _last)
6	priority_queue(_InputIterator _first, _InputIterator _last, const _Compare& _x)
7	priority_queue(_InputIterator _first, _InputIterator _last, const _Compare& _x, const _Sequence& _s)
8	bool empty()
9	size_type size()
10	const_reference top()
11	void push(const value_type& _x)
12	void pop()

15. <set>

<set>でサポートしているメソッドは以下になります。

set クラスのメソッド一覧	
1	set(const _Compare& __comp = _Compare(),const allocator_type& __a = allocator_type())
2	set(_InputIterator __first, _InputIterator __last)
3	set(_InputIterator __first, _InputIterator __last, const _Compare& __comp,const allocator_type& __a = allocator_type())
4	set(const _Self& __x)
5	set(__move_source<_Self> src)
6	key_compare key_comp()
7	value_compare value_comp()
8	allocator_type get_allocator()
9	iterator begin()
10	const_iterator begin()
11	iterator end()
12	const_iterator end()
13	reverse_iterator rbegin()
14	const_reverse_iterator rbegin()
15	reverse_iterator rend()
16	const_reverse_iterator rend()
17	bool empty()
18	size_type size()
19	size_type max_size()
20	swap(_Self& __x)
21	pair<iterator,bool> insert(const value_type& __x)
22	iterator insert(iterator __pos, const value_type& __x)
23	void insert(_InputIterator __first, _InputIterator __last)
24	void erase(iterator __pos)
25	size_type erase(const key_type& __x)
26	void erase(iterator __first, iterator __last)
27	clear()
28	const_iterator find(const _KT& __x)
29	iterator find(const _KT& __x)
30	size_type count(const _KT& __x)
31	iterator lower_bound(const _KT& __x)
32	const_iterator lower_bound(const _KT& __x)
33	iterator upper_bound(const _KT& __x)
34	const_iterator upper_bound(const _KT& __x)
35	pair<iterator, iterator> equal_range(const _KT& __x)
36	pair<const_iterator, const_iterator> equal_range(const _KT& __x)
37	_Self& operator=(const _Self& __x)
38	bool _STLP_CALL operator!= (const set<_Key,_Compare,_Alloc>& __x, const set<_Key,_Compare,_Alloc>& __y)
39	bool _STLP_CALL operator< (const set<_Key,_Compare,_Alloc>& __x, const set<_Key,_Compare,_Alloc>& __y)
40	bool _STLP_CALL operator<= (const set<_Key,_Compare,_Alloc>& __x, const set<_Key,_Compare,_Alloc>& __y)
41	bool _STLP_CALL operator== (const set<_Key,_Compare,_Alloc>& __x, const set<_Key,_Compare,_Alloc>& __y)
42	bool _STLP_CALL operator> (const set<_Key,_Compare,_Alloc>& __x, const set<_Key,_Compare,_Alloc>& __y)
43	bool _STLP_CALL operator>= (const set<_Key,_Compare,_Alloc>& __x, const set<_Key,_Compare,_Alloc>& __y)

multiset クラスのメソッド一覧	
1	multiset(const _Compare& _comp = _Compare(),const allocator_type& _a = allocator_type())
2	multiset(_InputIterator _first, _InputIterator _last)
3	multiset(_InputIterator _first, _InputIterator _last,const _Compare& _comp,const allocator_type& _a = allocator_type())
4	multiset(const _Self& _x)
5	multiset(_move_source<_Self> src)
6	key_compare key_comp()
7	value_compare value_comp()
8	allocator_type get_allocator()
9	iterator begin()
10	const_iterator begin()
11	iterator end()
12	const_iterator end()
13	reverse_iterator rbegin()
14	const_reverse_iterator rbegin()
15	reverse_iterator rend()
16	const_reverse_iterator rend()
17	bool empty()
18	size_type size()
19	size_type max_size()
20	void swap(_Self& _x)
21	iterator insert(const value_type& _x)
22	iterator insert(iterator _pos, const value_type& _x)
23	void insert(_InputIterator _first, _InputIterator _last)
24	void erase(iterator _pos)
25	size_type erase(const key_type& _x)
26	void erase(iterator _first, iterator _last)
27	void clear()
28	iterator find(const _KT& _x)
29	const_iterator find(const _KT& _x)
30	size_type count(const _KT& _x)
31	iterator lower_bound(const _KT& _x)
32	const_iterator lower_bound(const _KT& _x)
33	iterator upper_bound(const _KT& _x)
34	const_iterator upper_bound(const _KT& _x)
35	pair<iterator, iterator> equal_range(const _KT& _x)
36	pair<const_iterator, const_iterator> equal_range(const _KT& _x)
37	_Self& operator=(const _Self& _x)
38	bool _STLP_CALL operator!= (const multiset<_Key,_Compare,_Alloc>& _x, const multiset<_Key,_Compare,_Alloc>& _y)
39	bool _STLP_CALL operator< (const multiset<_Key,_Compare,_Alloc>& _x, const multiset<_Key,_Compare,_Alloc>& _y)
40	bool _STLP_CALL operator<= (const multiset<_Key,_Compare,_Alloc>& _x, const multiset<_Key,_Compare,_Alloc>& _y)
41	bool _STLP_CALL operator== (const multiset<_Key,_Compare,_Alloc>& _x, const multiset<_Key,_Compare,_Alloc>& _y)
42	bool _STLP_CALL operator> (const multiset<_Key,_Compare,_Alloc>& _x, const multiset<_Key,_Compare,_Alloc>& _y)
43	bool _STLP_CALL operator>= (const multiset<_Key,_Compare,_Alloc>& _x, const multiset<_Key,_Compare,_Alloc>& _y)

16. <stack>

<stack>でサポートしているメソッドは以下になります。

stack クラスのメソッド一覧	
1	stack()
2	stack(const _Sequence& _s)
3	stack(__move_source<_Self> src)
4	bool empty()
5	size_type size()
6	reference top()
7	const_reference top()
8	void push(const value_type& _x)
9	void pop()
10	const _Sequence& _Get_s()
11	bool _STLP_CALL operator!= (const stack<_Tp>& _x, const stack<_Tp>& _y)
12	bool _STLP_CALL operator< (const stack<_Tp>& _x, const stack<_Tp>& _y)
13	bool _STLP_CALL operator<= (const stack<_Tp>& _x, const stack<_Tp>& _y)
14	bool _STLP_CALL operator== (const stack<_Tp>& _x, const stack<_Tp>& _y)
15	bool _STLP_CALL operator> (const stack<_Tp>& _x, const stack<_Tp>& _y)
16	bool _STLP_CALL operator>= (const stack<_Tp>& _x, const stack<_Tp>& _y)

17. <stdexcept>

<stdexcept>でサポートしているメソッドは以下になります。

stdexcept クラスのメソッド一覧	
1	logic_error(const string& _s)
2	runtime_error(const string& _s)
3	domain_error(const string& _arg)
4	invalid_argument(const string& _arg)
5	length_error(const string& _arg)
6	out_of_range(const string& _arg)
7	range_error(const string& _arg)
8	overflow_error(const string& _arg)
9	underflow_error(const string& _arg)

18. <stlstring>

<stlstring>でサポートしているメソッドは以下になります。

stlstring クラスのメソッド一覧	
1	explicit basic_string(const allocator_type& _a = allocator_type())
2	basic_string(_Reserve_t, size_t _n, const allocator_type& _a = allocator_type())
3	basic_string(const _Self&)
4	basic_string(const _Self& _s, size_type _pos, size_type _n = npos, const allocator_type& _a = allocator_type())
5	basic_string(const _CharT* _s, size_type _n, const allocator_type& _a = allocator_type())
6	basic_string(const _CharT* _s, const allocator_type& _a = allocator_type());
7	basic_string(size_type _n, _CharT _c,　const allocator_type& _a = allocator_type())
8	basic_string(_move_source<_Self> src)
9	basic_string(_InputIterator _f, _InputIterator _l,const allocator_type & _a = allocator_type())
10	_Self& operator=(const _Self& _s)
11	_Self& operator=(const _CharT* _s)
12	_Self& operator=(_CharT _c)
13	iterator begin()
14	const_iterator begin()
15	iterator end()
16	const_iterator end()
17	reverse_iterator rbegin()
18	const_reverse_iterator rbegin()
19	reverse_iterator rend()
20	const_reverse_iterator rend()
21	size_type size()
22	size_type length()
23	size_type max_size()
24	void resize(size_type _n, _CharT _c)
25	void resize(size_type _n)
26	void reserve(size_type = 0)
27	size_type capacity()
28	void clear()
29	bool empty()
30	const_reference operator[](size_type _n)
31	reference operator[](size_type _n)
32	const_reference at(size_type _n)
33	reference at(size_type _n)
34	_Self& operator+=(const _Self& _s)
35	_Self& operator+=(const _CharT* _s)
36	_Self& operator+=(_CharT _c)
37	_Self& append(_InputIter _first, _InputIter _last)
38	_Self& append(const _Self& _s)
39	_Self& append(const _CharT* _s, size_type _n)
40	_Self& append(const _CharT* _s)
41	void push_back(_CharT _c)
42	void pop_back()
43	_Self& assign(const _Self& _s)
44	_Self& assign(const _Self& _s, size_type _pos, size_type _n)
45	_Self& assign(const _CharT* _s, size_type _n)
46	_Self& assign(const _CharT* _s)
47	_Self& assign(size_type _n, _CharT _c)
48	_Self& assign(_InputIter _first, _InputIter _last)
49	_Self& insert(size_type _pos, const _Self& _s)
50	_Self& insert(size_type _pos, const _Self& _s,size_type _beg, size_type _n)
51	_Self& insert(size_type _pos, const _CharT* _s, size_type _n)
52	_Self& insert(size_type _pos, const _CharT* _s)
53	_Self& insert(size_type _pos, size_type _n, _CharT _c)
54	iterator insert(iterator _p, _CharT _c)
55	void insert(iterator _p, size_t _n, _CharT _c)

56	void insert(iterator _p, _InputIter _first, _InputIter _last)
57	void insert(iterator _p, const _CharT* _f, const _CharT* _l)
58	_Self& erase(size_type _pos = 0, size_type _n = npos)
59	iterator erase(iterator _pos)
60	iterator erase(iterator _first, iterator _last)
61	_Self& replace(size_type _pos, size_type _n, const _Self& _s)
62	_Self& replace(size_type _pos1, size_type _n1, const _Self& _s, size_type _pos2, size_type _n2)
63	_Self& replace(size_type _pos, size_type _n1, const _CharT* _s, size_type _n2)
64	_Self& replace(size_type _pos, size_type _n1, const _CharT* _s)
65	_Self& replace(size_type _pos, size_type _n1,size_type _n2, _CharT _c)
66	_Self& replace(iterator _first, iterator _last, const _Self& _s)
67	_Self& replace(iterator _first, iterator _last,const _CharT* _s)
68	_Self& replace(iterator _first, iterator _last,_InputIter _f, _InputIter _l)
69	Self& replace(iterator _first, iterator _last,const _CharT* _f, const _CharT* _l)
70	size_type copy(_CharT* _s, size_type _n, size_type _pos = 0)
71	void swap(_Self& _s)
72	const _CharT* c_str()
73	const _CharT* data()
74	size_type find(const _Self& _s, size_type _pos = 0)
75	size_type find(const _CharT* _s, size_type _pos = 0)
76	size_type find(const _CharT* _s, size_type _pos, size_type _n)
77	size_type find(_CharT _c)
78	size_type find(_CharT _c, size_type _pos /* = 0 */)
79	size_type rfind(const _Self& _s, size_type _pos = npos)
80	size_type rfind(const _CharT* _s, size_type _pos = npos)
81	size_type rfind(const _CharT* _s, size_type _pos, size_type _n)
82	size_type rfind(_CharT _c, size_type _pos = npos)
83	size_type find_first_of(const _Self& _s, size_type _pos = 0)
84	size_type find_first_of(const _CharT* _s, size_type _pos = 0)
85	size_type find_first_of(const _CharT* _s, size_type _pos, size_type _n)
86	size_type find_first_of(_CharT _c, size_type _pos = 0)
87	size_type find_last_of(const _Self& _s, size_type _pos = npos)
88	size_type find_last_of(const _CharT* _s, size_type _pos = npos)
89	size_type find_last_of(const _CharT* _s, size_type _pos, size_type _n)
90	size_type find_last_of(_CharT _c, size_type _pos = npos)
91	size_type find_first_not_of(const _Self& _s, size_type _pos = 0)
92	size_type find_first_not_of(const _CharT* _s, size_type _pos = 0)
93	size_type find_first_not_of(const _CharT* _s, size_type _pos, size_type _n)
94	size_type find_first_not_of(_CharT _c, size_type _pos = 0)
95	size_type find_last_not_of(const _Self& _s, size_type _pos = npos)
96	size_type find_last_not_of(const _CharT* _s, size_type _pos = npos)
97	size_type find_last_not_of(const _CharT* _s, size_type _pos, size_type _n)
98	size_type find_last_not_of(_CharT _c, size_type _pos = npos)
99	_Self substr(size_type _pos = 0, size_type _n = npos)
100	int compare(const _Self& _s)
101	int compare(size_type _pos1, size_type _n1, const _Self& _s)
102	int compare(size_type _pos1, size_type _n1, const _Self& _s,　size_type _pos2, size_type _n2)
103	int compare(const _CharT* _s)
104	int compare(size_type _pos1, size_type _n1, const _CharT* _s)
105	int compare(size_type _pos1, size_type _n1, const _CharT* _s, size_type _n2)
106	static int　_M_compare(const _CharT* _f1, const _CharT* _l1, const _CharT* _f2, const _CharT* _l2)
107	basic_string<_CharT,_Traits,_Alloc> operator+(const basic_string<_CharT,_Traits,_Alloc>& _s, const basic_string<_CharT,_Traits,_Alloc>& _y)
108	basic_string<_CharT,_Traits,_Alloc> operator+(const _CharT* _s,const basic_string<_CharT,_Traits,_Alloc>& _y)
109	basic_string<_CharT,_Traits,_Alloc> operator+(_CharT _c, const basic_string<_CharT,_Traits,_Alloc>& _y)
110	basic_string<_CharT,_Traits,_Alloc> operator+(const basic_string<_CharT,_Traits,_Alloc>& _x, const _CharT* _s)
111	basic_string<_CharT,_Traits,_Alloc> operator+(const basic_string<_CharT,_Traits,_Alloc>& _x, const _CharT _c)
112	bool _STLP_CALL operator==(const _CharT* _s, const basic_string<_CharT,_Traits,_Alloc>& _y)
113	bool _STLP_CALL operator==(const basic_string<_CharT,_Traits,_Alloc>& _x,const _CharT* _s)
114	bool _STLP_CALL operator<(const basic_string<_CharT,_Traits,_Alloc>& _x,const basic_string<_CharT,_Traits,_Alloc>& _y)
115	bool _STLP_CALL operator<(const basic_string<_CharT,_Traits,_Alloc>& _x, const _CharT* _s)
116	bool _STLP_CALL operator!=(const basic_string<_CharT,_Traits,_Alloc>& _x, const basic_string<_CharT,_Traits,_Alloc>& _y)

117	bool _STLP_CALL operator>(const basic_string<_CharT,_Traits,_Alloc>& _x, const basic_string<_CharT,_Traits,_Alloc>& _y)
118	bool _STLP_CALL operator<=(const basic_string<_CharT,_Traits,_Alloc>& _x, const basic_string<_CharT,_Traits,_Alloc>& _y)
119	bool _STLP_CALL operator>=(const basic_string<_CharT,_Traits,_Alloc>& _x, const basic_string<_CharT,_Traits,_Alloc>& _y)
120	bool _STLP_CALL operator!=(const _CharT* _s,const basic_string<_CharT,_Traits,_Alloc>& _y)
121	bool _STLP_CALL operator!=(const basic_string<_CharT,_Traits,_Alloc>& _x, const _CharT* _s)
122	bool _STLP_CALL operator>(const _CharT* _s, const basic_string<_CharT,_Traits,_Alloc>& _y)
123	bool _STLP_CALL operator>(const basic_string<_CharT,_Traits,_Alloc>& _x, const _CharT* _s)
124	bool _STLP_CALL operator<=(const _CharT* _s, const basic_string<_CharT,_Traits,_Alloc>& _y)
125	bool _STLP_CALL operator<=(const basic_string<_CharT,_Traits,_Alloc>& _x,　const _CharT* _s)
126	bool _STLP_CALL operator>=(const _CharT* _s, const basic_string<_CharT,_Traits,_Alloc>& _y)
127	bool _STLP_CALL operator>=(const basic_string<_CharT,_Traits,_Alloc>& _x, const _CharT* _s)

19. <typeinfo>

<typeinfo>でサポートしているメソッドは以下になります。

typeinfo クラスのメソッド一覧	
1	~type_info()
2	_bool operator==(const type_info&)
3	_bool operator!=(const type_info&)
4	_bool before(const type_info&)
5	const char* name()

20. <utility>

<utility>でサポートしているメソッドは以下になります。

utility クラスのメソッド一覧	
1	pair()
2	pair(const _T1& _a, const _T2& _b)
3	pair(const pair<_U1, _U2>& _p)
4	pair(const pair<_T1,_T2>& _o)
5	pair(__move_source<pair<_T1, _T2> > src)
6	pair<_T1, _T2 const*> make_pair(_T1 const& _x,_T2 const (&_y)[_Sz])
7	pair<_T1 const*, _T2> make_pair(_T1 const (&_x)[_Sz],_T2 const& _y)
8	pair<_T1 const*, _T2 const*> make_pair(_T1 const (&_x)[_Sz1],_T2 const (&_y)[_Sz2])
9	pair<_T1, _T2>　make_pair(_T1 _x, _T2 _y)
10	bool _STLP_CALL operator== (const pair<_T1, _T2>& _x, const pair<_T1, _T2>& _y)
11	bool _STLP_CALL operator< (const pair<_T1, _T2>& _x, const pair<_T1, _T2>& _y)
12	bool _STLP_CALL operator!= (const pair<_T1, _T2>& _x, const pair<_T1, _T2>& _y)
13	bool _STLP_CALL operator> (const pair<_T1, _T2>& _x, const pair<_T1, _T2>& _y)
14	bool _STLP_CALL operator<= (const pair<_T1, _T2>& _x, const pair<_T1, _T2>& _y)
15	bool _STLP_CALL operator>= (const pair<_T1, _T2>& _x, const pair<_T1, _T2>& _y)

21. <valarray>

<valarray>でサポートしているメソッドは以下になります。

valarray クラスのメソッド一覧	
1	valarray()
2	explicit valarray(size_t _n)
3	valarray(const value_type& _x, size_t _n)
4	valarray(const value_type* _p, size_t _n)
5	valarray(const valarray<_Tp>& _x)
6	valarray(const slice_array<_Tp>&)
7	valarray(const gslice_array<_Tp>&)
8	valarray(const mask_array<_Tp>&)
9	valarray(const indirect_array<_Tp>&)
10	~valarray()
11	valarray<_Tp>& operator=(const valarray<_Tp>& _x)
12	valarray<_Tp>& operator=(const value_type& _x)
13	valarray<_Tp>& operator=(const slice_array<_Tp>&)
14	valarray<_Tp>& operator=(const gslice_array<_Tp>&)
15	valarray<_Tp>& operator=(const mask_array<_Tp>&)
16	valarray<_Tp>& operator=(const indirect_array<_Tp>&)
17	value_type operator[](size_t _n)
18	value_type& operator[](size_t _n)
19	valarray<_Tp> operator[](slice) const
20	slice_array<_Tp> operator[](slice)
21	valarray<_Tp> operator[](const gslice&)
22	gslice_array<_Tp> operator[](const gslice&)
23	valarray<_Tp> operator[](const _Valarray_bool&)
24	mask_array<_Tp> operator[](const _Valarray_bool&)
25	valarray<_Tp> operator[](const _Valarray_size_t&)
26	indirect_array<_Tp> operator[](const _Valarray_size_t&)
27	size_t size()
28	valarray<_Tp> operator+()
29	valarray<_Tp> operator-()
30	valarray<_Tp> operator~()
31	_Valarray_bool operator!()
32	valarray<_Tp>& operator*= (const value_type& _x)
33	valarray<_Tp>& operator*= (const valarray<_Tp>& _x)
34	valarray<_Tp>& operator/= (const value_type& _x)
35	valarray<_Tp>& operator/= (const valarray<_Tp>& _x)
36	valarray<_Tp>& operator%= (const value_type& _x)
37	valarray<_Tp>& operator%= (const valarray<_Tp>& _x)
38	valarray<_Tp>& operator+= (const value_type& _x)
39	valarray<_Tp>& operator+= (const valarray<_Tp>& _x)
40	valarray<_Tp>& operator-= (const value_type& _x)
41	valarray<_Tp>& operator-= (const valarray<_Tp>& _x)
42	valarray<_Tp>& operator^= (const value_type& _x)
43	valarray<_Tp>& operator^= (const valarray<_Tp>& _x)
44	valarray<_Tp>& operator&= (const value_type& _x)
45	valarray<_Tp>& operator&= (const valarray<_Tp>& _x)
46	valarray<_Tp>& operator|= (const value_type& _x)
47	valarray<_Tp>& operator|= (const valarray<_Tp>& _x)
48	valarray<_Tp>& operator<<= (const value_type& _x)
49	valarray<_Tp>& operator<<= (const valarray<_Tp>& _x)
50	valarray<_Tp>& operator>>= (const value_type& _x)
51	valarray<_Tp>& operator>>= (const valarray<_Tp>& _x)
52	value_type sum()
53	value_type (min) ()
54	value_type (max) ()
55	valarray<_Tp> shift(int _n)

56	valarray<_Tp> cshift(int _n)
57	valarray<_Tp> apply(value_type _f(value_type))
58	valarray<_Tp> apply(value_type _f(const value_type&))
59	void resize(size_t _n, value_type _x = value_type())
60	valarray<_Tp>　_STLP_CALL operator*(const valarray<_Tp>& _x, const valarray<_Tp>& _y)
61	valarray<_Tp>　_STLP_CALL operator/(const valarray<_Tp>& _x, const valarray<_Tp>& _y)
62	valarray<_Tp>　_STLP_CALL operator%(const valarray<_Tp>& _x, const valarray<_Tp>& _y)
63	valarray<_Tp>　_STLP_CALL operator+(const valarray<_Tp>& _x, const valarray<_Tp>& _y)
64	valarray<_Tp>　_STLP_CALL operator−(const valarray<_Tp>& _x,　const valarray<_Tp>& _y)
65	valarray<_Tp> _STLP_CALL operator^(const valarray<_Tp>& _x, const valarray<_Tp>& _y)
66	valarray<_Tp> _STLP_CALL operator&(const valarray<_Tp>& _x, const valarray<_Tp>& _y)
67	valarray<_Tp> _STLP_CALL operator|(const valarray<_Tp>& _x, const valarray<_Tp>& _y)
68	valarray<_Tp> _STLP_CALL operator<<(const valarray<_Tp>& _x, const valarray<_Tp>& _y)
69	valarray<_Tp> _STLP_CALL operator>>(const valarray<_Tp>& _x, const valarray<_Tp>& _y)
70	valarray<_Tp> _STLP_CALL operator*(const valarray<_Tp>& _x, const _Tp& _c)
71	valarray<_Tp> _STLP_CALL operator*(const _Tp& _c, const valarray<_Tp>& _x)
72	valarray<_Tp> _STLP_CALL operator/(const valarray<_Tp>& _x, const _Tp& _c)
73	valarray<_Tp> _STLP_CALL operator/(const _Tp& _c, const valarray<_Tp>& _x)
74	valarray<_Tp> _STLP_CALL operator%(const valarray<_Tp>& _x, const _Tp& _c)
75	valarray<_Tp> _STLP_CALL operator%(const _Tp& _c, const valarray<_Tp>& _x)
76	valarray<_Tp> _STLP_CALL operator+(const valarray<_Tp>& _x, const _Tp& _c)
77	valarray<_Tp> _STLP_CALL operator+(const _Tp& _c, const valarray<_Tp>& _x)
78	valarray<_Tp> _STLP_CALL operator−(const valarray<_Tp>& _x, const _Tp& _c)
79	valarray<_Tp> _STLP_CALL operator−(const _Tp& _c, const valarray<_Tp>& _x)
80	valarray<_Tp> _STLP_CALL operator^(const valarray<_Tp>& _x, const _Tp& _c)
81	valarray<_Tp> _STLP_CALL operator^(const _Tp& _c, const valarray<_Tp>& _x)
82	valarray<_Tp> _STLP_CALL operator&(const valarray<_Tp>& _x, const _Tp& _c)
83	valarray<_Tp> _STLP_CALL operator&(const _Tp& _c, const valarray<_Tp>& _x)
84	valarray<_Tp> _STLP_CALL operator|(const valarray<_Tp>& _x, const _Tp& _c)
85	valarray<_Tp> _STLP_CALL operator|(const _Tp& _c, const valarray<_Tp>& _x)
86	valarray<_Tp> _STLP_CALL operator<<(const valarray<_Tp>& _x, const _Tp& _c)
87	valarray<_Tp> _STLP_CALL operator<<(const _Tp& _c, const valarray<_Tp>& _x)
88	valarray<_Tp> _STLP_CALL operator>>(const valarray<_Tp>& _x, const _Tp& _c)
89	valarray<_Tp> _STLP_CALL operator>>(const _Tp& _c, const valarray<_Tp>& _x)
90	_Valarray_bool _STLP_CALL operator==(const valarray<_Tp>& _x, const valarray<_Tp>& _y)
91	_Valarray_bool _STLP_CALL operator>(const valarray<_Tp>& _x, const valarray<_Tp>& _y)
92	_Valarray_bool _STLP_CALL operator!=(const valarray<_Tp>& _x,　const valarray<_Tp>& _y)
93	_Valarray_bool _STLP_CALL operator>(const valarray<_Tp>& _x,　const valarray<_Tp>& _y)
94	_Valarray_bool _STLP_CALL operator<=(const valarray<_Tp>& _x, const valarray<_Tp>& _y)
95	_Valarray_bool _STLP_CALL operator>=(const valarray<_Tp>& _x,　const valarray<_Tp>& _y)
96	_Valarray_bool _STLP_CALL operator&&(const valarray<_Tp>& _x, const valarray<_Tp>& _y)
97	_Valarray_bool _STLP_CALL operator||(const valarray<_Tp>& _x, const valarray<_Tp>& _y)
98	_Valarray_bool _STLP_CALL operator==(const valarray<_Tp>& _x, const _Tp& _c)
99	_Valarray_bool _STLP_CALL operator==(const _Tp& _c, const valarray<_Tp>& _x)
100	_Valarray_bool _STLP_CALL operator!=(const valarray<_Tp>& _x, const _Tp& _c)
101	_Valarray_bool _STLP_CALL operator!=(const _Tp& _c, const valarray<_Tp>& _x)
102	_Valarray_bool _STLP_CALL operator<(const valarray<_Tp>& _x, const _Tp& _c)
103	_Valarray_bool _STLP_CALL operator<(const _Tp& _c, const valarray<_Tp>& _x)
104	_Valarray_bool _STLP_CALL operator>(const valarray<_Tp>& _x, const _Tp& _c)
105	_Valarray_bool _STLP_CALL operator>(const _Tp& _c, const valarray<_Tp>& _x)
106	_Valarray_bool _STLP_CALL operator<=(const valarray<_Tp>& _x, const _Tp& _c)
107	_Valarray_bool _STLP_CALL operator<=(const valarray<_Tp>& _x, const _Tp& _c)
108	_Valarray_bool _STLP_CALL operator>=(const valarray<_Tp>& _x, const _Tp& _c)
109	_Valarray_bool _STLP_CALL operator>=(const _Tp& _c, const valarray<_Tp>& _x)
110	_Valarray_bool _STLP_CALL operator&&(const valarray<_Tp>& _x, const _Tp& _c)
111	_Valarray_bool _STLP_CALL operator&&(const _Tp& _c, const valarray<_Tp>& _x)
112	_Valarray_bool _STLP_CALL operator||(const valarray<_Tp>& _x, const _Tp& _c)
113	_Valarray_bool _STLP_CALL operator||(const _Tp& _c, const valarray<_Tp>& _x)
114	valarray<_Tp> abs(const valarray<_Tp>& _x)
115	valarray<_Tp> acos(const valarray<_Tp>& _x)
116	valarray<_Tp> asin(const valarray<_Tp>& _x)
117	valarray<_Tp> atan(const valarray<_Tp>& _x)

118	valarray<_Tp> atan2(const valarray<_Tp>& _x, const valarray<_Tp>& _y)
119	valarray<_Tp> atan2(const valarray<_Tp>& _x, const _Tp& _c)
120	valarray<_Tp> atan2(const _Tp& _c, const valarray<_Tp>& _x)
121	valarray<_Tp> cos(const valarray<_Tp>& _x)
122	valarray<_Tp> cosh(const valarray<_Tp>& _x)
123	valarray<_Tp> exp(const valarray<_Tp>& _x)
124	valarray<_Tp> log(const valarray<_Tp>& _x)
125	valarray<_Tp> log10(const valarray<_Tp>& _x)
126	valarray<_Tp> pow(const valarray<_Tp>& _x, const valarray<_Tp>& _y)
127	valarray<_Tp> pow(const valarray<_Tp>& _x, const _Tp& _c)
128	valarray<_Tp> pow(const _Tp& _c, const valarray<_Tp>& _x)
129	valarray<_Tp> sin(const valarray<_Tp>& _x)
130	valarray<_Tp> sinh(const valarray<_Tp>& _x)
131	valarray<_Tp> sqrt(const valarray<_Tp>& _x)
132	valarray<_Tp> tan(const valarray<_Tp>& _x)
133	valarray<_Tp> tanh(const valarray<_Tp>& _x)
134	slice()
135	slice(size_t _start, size_t _length, size_t _stride)
136	size_t start()
137	size_t size()
138	size_t stride()
139	void operator=(const valarray<value_type>& _x)
140	void operator=(const value_type& _c)
141	void operator*=(const valarray<value_type>& _x)
142	void operator/=(const valarray<value_type>& _x)
143	void operator%=(const valarray<value_type>& _x)
144	void operator+=(const valarray<value_type>& _x)
145	void operator-=(const valarray<value_type>& _x)
146	void operator^=(const valarray<value_type>& _x)
147	void operator&=(const valarray<value_type>& _x)
148	void operator|=(const valarray<value_type>& _x)
149	void operator<<=(const valarray<value_type>& _x)
150	void operator>>=(const valarray<value_type>& _x)
151	slice_array(const slice_array &_x)
152	~slice_array()
153	gslice()
154	gslice(size_t _start, const _Valarray_size_t& _lengths, const _Valarray_size_t& _strides)
155	start()
156	_Valarray_size_t size()
157	_Valarray_size_t stride()
158	bool _M_empty()
159	size_t _M_size()
160	void operator= (const valarray<value_type>& _x)
161	void operator= (const value_type& _c)
162	void operator*= (const valarray<value_type>& _x)
163	void operator/= (const valarray<value_type>& _x)
164	void operator%= (const valarray<value_type>& _x)
165	void operator+= (const valarray<value_type>& _x)
166	void operator-= (const valarray<value_type>& _x)
167	void operator^= (const valarray<value_type>& _x)
168	void operator&= (const valarray<value_type>& _x)
169	void operator|= (const valarray<value_type>& _x)
170	void operator<<= (const valarray<value_type>& _x)
171	void operator>>= (const valarray<value_type>& _x
172	gslice_array(const gslice_array& _x)
173	~gslice_array()
174	void operator=(const valarray<value_type>& _x)
175	void operator=(const value_type& _c)
176	void operator*=(const valarray<value_type>& _x)
177	void operator/=(const valarray<value_type>& _x)
178	void operator%=(const valarray<value_type>& _x)
179	void operator+=(const valarray<value_type>& _x)

180	void operator-=(const valarray<value_type>& _x)
181	void operator^=(const valarray<value_type>& _x)
182	void operator&=(const valarray<value_type>& _x)
183	void operator|=(const valarray<value_type>& _x)
184	void operator<<=(const valarray<value_type>& _x)
185	void operator>>=(const valarray<value_type>& _x)
186	mask_array(const mask_array& _x)
187	~mask_array()
188	void operator=(const valarray<value_type>& _x)
189	void operator=(const value_type& _c)
190	void operator*=(const valarray<value_type>& _x)
191	void operator/=(const valarray<value_type>& _x)
192	void operator%=(const valarray<value_type>& _x)
193	void operator+=(const valarray<value_type>& _x)
194	void operator-=(const valarray<value_type>& _x)
195	void operator^=(const valarray<value_type>& _x)
196	void operator&=(const valarray<value_type>& _x)
197	void operator|=(const valarray<value_type>& _x)
198	void operator<<=(const valarray<value_type>& _x)
199	void operator>>=(const valarray<value_type>& _x)
200	indirect_array(const indirect_array& _x)
201	~indirect_array()

22. <vector>

<vector>でサポートしているメソッドは以下になります。

vector クラスのメソッド一覧	
1	allocator_type get_allocator()
2	iterator begin()
3	const_iterator begin()
4	iterator end()
5	const_iterator end()
6	reverse_iterator rbegin()
7	const_reverse_iterator rbegin()
8	reverse_iterator rend()
9	const_reverse_iterator rend()
10	size_type size()
11	size_type max_size()
12	size_type capacity()
13	bool empty()
14	reference front()
15	const_reference front()
16	reference back()
17	const_reference back()
18	reference at(size_type _n)
19	const_reference at(size_type _n)
20	vector(const allocator_type& _a = allocator_type())
21	vector(size_type _n)
22	vector(const _Self& _x)
23	vector(_move_source<_Self> src)
24	vector(_InputIterator _first, _InputIterator _last,const allocator_type& _a = allocator_type())
25	void reserve(size_type _n)
26	void assign(size_type _n, const _Tp& _val)
27	void _M_fill_assign(size_type _n, const _Tp& _val)
28	void push_back(const _Tp& _x = _Tp())
29	iterator insert(iterator _pos, const _Tp& _x = _Tp())
30	void insert(iterator _pos, _InputIterator _first, _InputIterator _last)
31	void insert (iterator _pos, size_type _n, const _Tp& _x)
32	void swap(_Self& _x)
33	void pop_back()
34	iterator erase(iterator _pos)
35	iterator erase(iterator _first, iterator _last)
36	void resize(size_type _new_size, const _Tp& _x = _STLP_DEFAULT_CONSTRUCTED(_Tp))
37	void clear()
38	reference operator[] (size_type _n)
39	_Self& operator= (const _Self& _x)
40	bool _STLP_CALL operator== (const vector<_Tp>& _x, const vector<_Tp>& _y)
41	bool _STLP_CALL operator< (const vector<_Tp>& _x, const vector<_Tp>& _y)
42	bool _STLP_CALL operator!= (const vector<_Tp>& _x, const vector<_Tp>& _y)
43	bool _STLP_CALL operator<= (const vector<_Tp>& _x, const vector<_Tp>& _y)
44	bool _STLP_CALL operator> (const vector<_Tp>& _x, const vector<_Tp>& _y)
45	bool _STLP_CALL operator>= (const vector<_Tp>& _x, const vector<_Tp>& _y)

ホームページとサポート窓口

ルネサス エレクトロニクスホームページ

http://japan.renesas.com/

お問合せ先

http://japan.renesas.com/inquiry

改訂記録

Rev.	発行日	改訂内容	
		ページ	ポイント
1.00	2010.07.30	—	初版発行
-	-	-	-
-	-	-	-
-	-	-	-
-	-	-	-
-	-	-	-
-	-	-	-
-	-	-	-

ご注意書き

1. 本資料に記載されている内容は本資料発行時点のものであり、予告なく変更することがあります。当社製品のご購入およびご使用にあたりましては、事前に当社営業窓口で最新の情報をご確認いただきますとともに、当社ホームページなどを通じて公開される情報に常にご注意ください。
2. 本資料に記載された当社製品および技術情報の使用に関連し発生した第三者の特許権、著作権その他の知的財産権の侵害等に関し、当社は、一切その責任を負いません。当社は、本資料に基づき当社または第三者の特許権、著作権その他の知的財産権を何ら許諾するものではありません。
3. 当社製品を改造、改変、複製等しないでください。
4. 本資料に記載された回路、ソフトウェアおよびこれらに関連する情報は、半導体製品の動作例、応用例を説明するものです。お客様の機器の設計において、回路、ソフトウェアおよびこれらに関連する情報を使用する場合には、お客様の責任において行ってください。これらの使用に起因しお客様または第三者に生じた損害に関し、当社は、一切その責任を負いません。
5. 輸出に際しては、「外国為替及び外国貿易法」その他輸出関連法令を遵守し、かかる法令の定めるところにより必要な手続を行ってください。本資料に記載されている当社製品および技術を大量破壊兵器の開発等の目的、軍事利用の目的その他軍事用途の目的で使用しないでください。また、当社製品および技術を国内外の法令および規則により製造・使用・販売を禁止されている機器に使用することができません。
6. 本資料に記載されている情報は、正確を期すため慎重に作成したものですが、誤りがないことを保証するものではありません。万一、本資料に記載されている情報の誤りに起因する損害がお客様に生じた場合においても、当社は、一切その責任を負いません。
7. 当社は、当社製品の品質水準を「標準水準」、「高品質水準」および「特定水準」に分類しております。また、各品質水準は、以下に示す用途に製品が使われることを意図しておりますので、当社製品の品質水準をご確認ください。お客様は、当社の文書による事前の承諾を得ることなく、「特定水準」に分類された用途に当社製品を使用することができません。また、お客様は、当社の文書による事前の承諾を得ることなく、意図されていない用途に当社製品を使用することができません。当社の文書による事前の承諾を得ることなく、「特定水準」に分類された用途または意図されていない用途に当社製品を使用したことによりお客様または第三者に生じた損害等に関し、当社は、一切その責任を負いません。なお、当社製品のデータ・シート、データ・ブック等の資料で特に品質水準の表示がない場合は、標準水準製品であることを表します。

   標準水準： コンピュータ、OA機器、通信機器、計測機器、AV機器、家電、工作機械、パーソナル機器、産業用ロボット

   高品質水準： 輸送機器（自動車、電車、船舶等）、交通用信号機器、防災・防犯装置、各種安全装置、生命維持を目的として設計されていない医療機器（厚生労働省定義の管理医療機器に相当）

   特定水準： 航空機器、航空宇宙機器、海底中継機器、原子力制御システム、生命維持のための医療機器（生命維持装置、人体に埋め込み使用するもの、治療行為（患部切り出し等）を行うもの、その他直接人命に影響を与えるもの）（厚生労働省定義の高度管理医療機器に相当）またはシステム等
8. 本資料に記載された当社製品のご使用につき、特に、最大定格、動作電源電圧範囲、放熱特性、実装条件その他諸条件につきましては、当社保証範囲内でご使用ください。当社保証範囲を超えて当社製品をご使用された場合の故障および事故につきましては、当社は、一切その責任を負いません。
9. 当社は、当社製品の品質および信頼性の向上に努めておりますが、半導体製品はある確率で故障が発生したり、使用条件によっては誤動作したりする場合があります。また、当社製品は耐放射線設計については行っておりません。当社製品の故障または誤動作が生じた場合も、人身事故、火災事故、社会的損害などを生じさせないようお客様の責任において冗長設計、延焼対策設計、誤動作防止設計等の安全設計およびエージング処理等、機器またはシステムとしての出荷保証をお願いいたします。特に、マイコンソフトウェアは、単独での検証は困難なため、お客様が製造された最終の機器・システムとしての安全検証をお願いいたします。
10. 当社製品の環境適合性等、詳細につきましては製品個別に必ず当社営業窓口までお問合せください。ご使用に際しては、特定の物質の含有･使用を規制するRoHS指令等、適用される環境関連法令を十分調査のうえ、かかる法令に適合するようご使用ください。お客様がかかる法令を遵守しないことにより生じた損害に関して、当社は、一切その責任を負いません。
11. 本資料の全部または一部を当社の文書による事前の承諾を得ることなく転載または複製することを固くお断りいたします。
12. 本資料に関する詳細についてのお問い合わせその他お気付きの点等がございましたら当社営業窓口までご照会ください。

注1. 本資料において使用されている「当社」とは、ルネサス エレクトロニクス株式会社およびルネサス エレクトロニクス株式会社がその総株主の議決権の過半数を直接または間接に保有する会社をいいます。

注2. 本資料において使用されている「当社製品」とは、注1において定義された当社の開発、製造製品をいいます。

ルネサス エレクトロニクス株式会社

■営業お問合せ窓口 http://www.renesas.com

※営業お問合せ窓口の住所・電話番号は変更になることがあります。最新情報につきましては、弊社ホームページをご覧ください。

ルネサス エレクトロニクス販売株式会社　〒100-0004　千代田区大手町2-6-2（日本ビル） (03)5201-5307

■技術的なお問合せおよび資料のご請求は下記へどうぞ。
総合お問合せ窓口：http://japan.renesas.com/inquiry