RX ファミリ

M3S-TFS-Tiny:オリジナルファイルシステムソフトウェア

R20AN0039JJ0100
Rev.1.00
2010.10.08

要旨

本アプリケーションノートでは、サンプルプログラムと TFS ファイルシステムソフトウェアライブラリの使用方法について説明しています。

動作確認デバイス

RX ファミリ

目次

1. ライブラリ仕様

Tiny ファイルシステムライブラリの主な仕様を以下に示します。

仕様	値
互換性のあるメディアサイズ	32 MB、64 MB、128 MB、256 MB、512 MB、1 GB
FAT ラッピング FAT タイプ	FAT16
複数ドライブサポート	対応（作業領域は、初期化時に設定する必要あり）
ディレクトリ	ルートディレクトリのみ
ディレクトリエントリ数	最大 65,534 ブロック（フォーマット時にディレクトリ領域サイズを設定し保存する）
ディレクトリエントリサイズ	128 バイト固定長
ファイル指定	ファイル番号（ファイル名は使用できない）
同時に開くことができるファイル数	複数（作業領域は、初期化時に設定する必要あり）
ファイルサイズ	可変（ブロック単位で割り当て）
ファイルあたりの割り当て可能なブロック数	4
ブロックサイズ	フォーマット時に 8 KB、16 KB、32 KB、64 KB、128 KB または 256 KB からブロックサイズを選択します。
ブロック制限	最大 65,534 ブロック
I/O バッファサイズ	64 バイト固定長（論理セクタ）
I/O バッファ数	少なくとも 1 個

2. ライブラリの型の定義

このセクションでは、ライブラリで使用する型の定義について詳細に説明します。

Datatype	Typedef
unsigned char	uint8_t
unsigned short	uint16_t
unsigned long	uint32_t
signed char	int8_t
signed short	int16_t
signed long	int32_t

3. 用語の説明

このセクションでは、TFS ライブラリに関連する用語について説明します。

3.1 論理セクタ／論理セクタ番号

TFS は、64 バイト固定長ブロックに分割されると仮定されるドライブに読み取り／書き込みを行います。この 64 バイト固定長ブロックは論理セクタと呼ばれます。各論理セクタは、ゼロから昇順に論理セクタ番号で識別されます。

3.2 ドライブ／ドライブ番号

TFS は、FAT ボリューム（DOS パーティションと類似する）がファイルシステムに格納されるドライブとして識別されます。TFS が複数のドライブを持つ場合は、追加のドライブは 0 から始まる番号で識別する必要があります。ドライブ番号はこのドライブ識別番号です。

4. ライブラリ構造

このセクションでは、ライブラリで使用する構造について詳細に説明します。

4.1 tfs_volume - ボリューム構造

説明

この構造はドライブ情報を保持するために使用します。必要な構造数は使用するドライブ数と同じです。たとえば、ドライブ数が 1 の場合、1 つのみの構造変数が必要です。ドライブ数が 2 の場合、2 つの構造が必要です。（以下同様）

この構造のメンバにはユーザプログラムから直接アクセスしないでください。ユーザプログラムは、使用するドライブ数と等しいアレイサイズの構造変数アレイを宣言するだけです。

構造

データ型	構造要素	説明
uint8_t	is_mounted	TFS 内部使用
uint8_t	drv	
uint16_t	rootents	
uint16_t	blocks	
uint16_t	bsize	
uint32_t	start	
uint32_t	vsize	
uint32_t	rsize	
uint32_t	hsize	
uint32_t	dsize	

4.2 tfs_file - ファイル構造

説明

この構造はファイル情報を保持するために使用します。必要な構造数は同時に開くファイル数と同じです。たとえば、一度に使用するファイル数が 1 つのみの場合、1 つのみの構造変数が必要です。一度に使用するファイル数が 2 つの場合、2 つの構造が必要になります。（以下同様）

この構造のメンバにはユーザプログラムから直接アクセスしないでください。ユーザプログラムは、同時に使用するファイル数と等しいアレイサイズの構造変数アレイを宣言するだけです。

構造

データ型	構造要素	説明
uint8_t	is_open	TFS 内部使用
uint8_t	id	
uint8_t	drv	
uint8_t	flags	
uint16_t	ent	
uint32_t	size	
uint32_t	ptr	

4.3 tfs_buff - バッファ構造

説明

この構造は論理セクタバッファ情報を保持するために使用します。

この構造のメンバにはユーザプログラムから直接アクセスしないでください。ユーザプログラムは、1 つのみの要素を持つバッファ構造変数アレイを宣言しなければなりません。必要なアレイ要素数は、使用するドライブ数またはファイル数にかかわらず 1 つのみです。

構造

データ型	構造要素	説明
uint8_t	cnt	TFS 内部使用
uint8_t	drv	
uint32_t	lsec	
uint8_t	buf[]	

4.4 tfs_config - ファイルシステム構成

説明

この構造は、ユーザの要件に従ってファイルシステム構成を設定するために使用します。希望する値を使用してこの構造を初期化し、tfs_init 関数を呼び出し、これらの値を設定する必要があります。

構造

データ型	構造要素	説明
uint16_t	drives	使用するドライブ数（≧1）
uint16_t	files	使用するファイル記述子の数、つまり同時に開くファイルの数（≧1）
uint16_t	buffs	使用する論理セクタバッファ数（≧1）
struct tfs_volume*	volume	ボリューム構造アレイの開始アドレス。 アレイ要素数は使用するドライブ数と同じでなければなりません。
struct tfs_file*	file	ファイル構造アレイの開始アドレス。 アレイ要素数は使用するファイル数と同じでなければなりません。
struct tfs_buff*	buff	バッファ構造アレイの開始アドレス。 このアレイには 1 つだけの要素があれば十分です。

4.5 tfs_format_param - FAT16 パラメータ

説明

この構造は tfs_format_param1 構造のメンバです。ドライブをフォーマットするときに使用する FAT16 パラメータを保持します。

構造

データ型	構造要素	説明
uint32_t	TotSec	ボリュームのセクタ合計数
uint16_t	SecPerTrk	トラックあたりのセクタ数
uint16_t	NumHeads	ヘッド合計数
const uint8_t*	VolLab	ボリュームラベル

4.6　tfs_format_param1 - ファイルシステムフォーマットパラメータ

説明

この構造はメモリドライブのフォーマッティングパラメータを保持します。

構造

データ型	構造要素	説明
struct tfs_format_param	fat	FAT16 パラメータ（4.5 に記載）
uint16_t	rootents	ルートディレクトリエントリ数
uint16_t	bsize	ブロックサイズ（KB）

メンバ

fat.TotSec

ボリュームのセクタ合計数を設定します（512 バイト／セクタ）。

fat.SecPerTrk

ドライブ上のトラックあたりのセクタ数を設定します（BIOS パラメータ）。

fat.NumHeads

ドライブ上のヘッド数を設定します。

fat.VolLab

FAT ボリュームラベルを設定します。NULL を設定すると、ドライブ上のラベル"NO␣NAME␣␣␣"トラックを使用します。

rootents

ルートディレクトリのエントリ数を設定します。4 の整数倍の値を設定します。

bsize

データブロックサイズ（KB）を設定します。有効な値は 8、16、32、64、128、256 です。

4.7　tfs_stat - ファイルステータス

説明

この構造は tfs_stati 関数から返されるファイル情報を保持します。

構造

データ型	構造要素	説明
uint32_t	st_size	ファイルサイズ
uint16_t	st_mdate	ファイルの最終変更日付
uint16_t	st_mtime	ファイルの最終変更時刻
uint16_t	st_mode	ファイルモード

メンバ

st_size

ファイルのサイズをバイト単位で格納します。

st_mdate

ファイルの変更日付を格納します。

bit15:9 - 1980 年からの年（0～127 の範囲の値）

bit8:5 - 月（1～12 の範囲の値）

bit4:0 - 日（1～31 の範囲の値）

st_mtime

ファイルの変更時刻またはディレクトリの作成時刻を格納します。

bit15:9 - 時間（0～23 の範囲の値）

bit8:5 - 分（0～59 の範囲の値）

bit4:0 - 秒は 2 秒間隔で表示されます。（0～29 の範囲の値は 0-58 として表示されます。）

st_mode

ファイルモードは、ファイルが通常のファイルか、ディレクトリかを示すために使用します。

4.8　tfs_statfs - ファイルシステムステータス

説明

この構造は tfs_statfs 関数から返されるファイルシステム情報を保持します。

構造

データ型	構造要素	説明
uint16_t	f_bsize	ブロックサイズ（KB）
uint16_t	f_blocks	ブロックの合計数
uint16_t	f_bfree	使用可能な空きブロック数
uint16_t	f_files	ルートディレクトリエントリの合計数
uint16_t	f_ffree	空きディレクトリエントリ数

5. ライブラリエラーコード

このセクションでは、ライブラリ関数から返されるエラーコードに対応するマクロの意味を説明します。

マクロ	値	意味
TFS_EPERM	1	操作は許可されません。
TFS_ENOENT	2	そのようなファイルまたはディレクトリはありません。
TFS_ESRCH	3	そのようなプロセスはありません。
TFS_EINTR	4	システムコールが中断されました。
TFS_EIO	5	I/O エラー
TFS_ENXIO	6	そのようなデバイスまたはアドレスはありません。
TFS_E2BIG	7	引数リストが長すぎます。
TFS_EBADF	9	ファイル番号が不正です。
TFS_EAGAIN	11	やり直してください。
TFS_ENOMEM	12	メモリ不足
TFS_EACCES	13	許可が拒否されました。
TFS_EFAULT	14	アドレスが不正です。
TFS_EBUSY	16	デバイスまたはリソースがビジーです。
TFS_EEXIST	17	ファイルが存在します。
TFS_EXDEV	18	クロスデバイスリンク（装置間リンク）
TFS_ENODEV	19	そのようなデバイスはありません。
TFS_ENOTDIR	20	ディレクトリではありません。
TFS_EISDIR	21	ディレクトリです。
TFS_EINVAL	22	無効な引数
TFS_ENFILE	23	ファイルテーブルオーバーフロー
TFS_EMFILE	24	開いているファイルが多すぎます。
TFS_EFBIG	27	ファイルが大きすぎます。
TFS_ENOSPC	28	デバイス上にスペースが残っていません。
TFS_EROFS	30	読み取り専用ファイルシステム
TFS_ERANGE	34	計算結果が正しくありません。
TFS_EDEADLK	35	リソースデッドロックが発生しました。
TFS_ENAMETOOLONG	36	ファイル名が長すぎます。
TFS_ENOLCK	37	レコードロックを使用することはできません。
TFS_ENOTEMPTY	39	ディレクトリが空でありません。
TFS_ETIMEDOUT	100	操作がタイムアウトしました。

6. ライブラリ関数

6.1 R_tfs_init

プロトタイプ

```
int16_t R_tfs_init （const struct tfs_config *config）
```

説明

この関数は、構造 tfs_config で指定された構成で TFS ライブラリを初期化します。この関数は、その他のライブラリ関数を呼び出す前に呼び出す必要があります。

引数

引数	型	説明
config	const struct tfs_config*	セクション 4.4 に記載するように希望する値を使用してこの構造を初期化してください。

戻り値

型	説明
int16_t	関数が正常に実行された場合、戻り値は 0 です。 関数がエラーで終了した場合、戻り値は-1 です。

使用例

```
struct tfs_volume volume[1];
struct tfs_file file[1];
struct tfs_buff buff[1];
struct tfs_config conf = {
    1,    //No. of drives
    1,    //No. of file descriptors
    1,    //No. of buffers
    volume,  //Start address of volume array
    file,    //Start address of file descriptor array
    buff     //Start address of buffer array
};
int16_t ret_val;
ret_val  = R_tfs_init（ &conf ）;
```

6.2　R_tfs_exit

プロトタイプ

```
int16_t R_tfs_exit （uint16_t force）
```

説明

この関数は、ライブラリの最後の処理です。ただし、この関数は、ドライブがマウントされていない場合にのみ呼び出すことができます。ドライブがマウントされているときに、この関数が呼び出された場合は、エラーが発生します。

通常、引数 force には値 0 が設定されます。0 以外の値を設定すると、関数は強制終了を実行します。この関数を呼び出した後、ライブラリを再び初期化せずに他の関数を呼び出すことはできません（R_tfs_init 関数を呼び出すことにより）。

引数

引数	型	説明
force	uint16_t	正常終了するには 0 を設定します。 強制終了するには、その他の値を設定します

戻り値

型	説明
int16_t	関数が正常に実行された場合、戻り値は 0 です。 関数がエラーで終了した場合、戻り値は-1 です。

使用例

```
int16_t ret_val;
// Other code before end processing
ret_val = R_tfs_exit（0）;
```

6.3　R_tfs_format1

プロトタイプ

```
int16_t R_tfs_format1 （uint16_t drv, const struct tfs_format_param1 *param）
```

説明

この関数は、構造 param に設定されたパラメータを使用してドライブ drv をフォーマットします。

ドライブがマウントされていない場合にのみフォーマットすることができます。ドライブがマウントされているときにこの関数が呼び出された場合は、エラーが発生します。また、フォーマット時に、開いているファイルはすべて閉じられます。

フォーマットは以下の順序で実行されます。

- 最初に、ボリューム全体が FAT16 ファイルシステムとしてフォーマットされます。
- 次に、TFS 領域が作成した FAT16 ファイルシステムに単一ファイルとして保存されます。
- 最後に、内部 TFS 領域がフォーマットされ初期化されます。

引数

引数	型	説明
drv	uint16_t	フォーマットするドライブの番号
param	const struct tfs_format_param1*	セクション 4.5 および 4.6 に記載するように希望する値を使用してこの構造を初期化してください。

戻り値

型	説明
int16_t	関数が正常に実行された場合、戻り値は 0 です。 関数がエラーで終了した場合、戻り値は-1 です。

使用例

```
const struct tfs_format_param1 test = {
{
 （unsigned long）64*1024*2,   /* Total no. of sectors （512B/sector） */
63,             /* Sectors per track */
255,            /* Number of heads */
"TINYFS      "            /* Volume label */
},
64,             /* No. of root directory entries */
128             /* Size of data block （KB） */
};
int16_t ret_val;
// Library initialization
ret_val = R_tfs_format1（ 0, &test ）;
```

6.4　R_tfs_attach

プロトタイプ

```
int16_t R_tfs_attach（uint16_t drv）
```

説明

この関数は、引数として渡されたドライブ番号 drv に TFS ボリュームをマウントします。

引数

引数	型	説明
drv	uint16_t	TFS ボリュームをマウントするドライブ番号

戻り値

型	説明
int16_t	関数が正常に実行された場合、戻り値は 0 です。 関数がエラーで終了した場合、戻り値は-1 です。

使用例

```
int16_t ret_val;
// Library initialization
ret_val = R_tfs_attach（0）;
```

6.5　R_tfs_detach

プロトタイプ

```
int16_t R_tfs_detach（uint16_t drv, uint16_t force）
```

説明

この関数は、引数として渡されたドライブ drv をアンマウントします。ドライブが使用中の場合は、アンマウントすることはできません。ドライブが使用中である場合は、関数はエラーを返します。

通常、引数 force には値 0 が設定されます。0 以外の値を設定すると、関数は強制アンマウントを実行します。

引数

引数	型	説明
drv	uint16_t	TFS ボリュームをアンマウントするドライブ番号
force	uint16_t	正常終了するには 0 を設定します。 強制終了するには、その他の値を設定します。

戻り値

型	説明
int16_t	関数が正常に実行された場合、戻り値は 0 です。 関数がエラーで終了した場合、戻り値は-1 です。

使用例

```
int16_t ret_val;
// Initialization
R_tfs_attach（0）;
// Processing
ret_val = R_tfs_detach（0,0）;
```

6.6　R_tfs_alloci

プロトタイプ

```
uint16_t R_tfs_alloci（uint16_t drv, uint16_t did, uint16_t fid）
```

説明

この関数は、引数として渡されたドライブ drv 上で fid より大きい最初の使用可能なファイル番号を返します。ファイル番号をディレクトリの先頭から取得する場合は、fid 値を 0 に設定します。ディレクトリ番号 did には値 0（ルートディレクトリ）を設定する必要があります。

引数

引数	型	説明
drv	uint16_t	ファイルを作成するドライブの番号
did	uint16_t	値 0（ルートディレクトリ）を設定しなければなりません。
fid	uint16_t	最初の使用可能なファイル番号を検索するファイル番号

戻り値

型	説明
int16_t	関数が正常に実行された場合、使用可能なファイル番号を返します。 エラーが発生した場合、値 TFS_NONUM を返します。

使用例

```
uint16_t file_no;
// Initialization and other processing
file_no = R_tfs_alloci（0,0,0）;
```

6.7　R_tfs_openi

プロトタイプ

int16_t R_tfs_openi（uint16_t drv, uint16_t did, uint16_t fid, int16_t flags）

説明

この関数は、ドライブ drv 上のファイル fid を開きます。ディレクトリ番号 did には値 0（ルートディレクトリ）を設定する必要があります。ファイルは、フラグの論理和の組み合わせを使用して異なるモードで開くことができます。

引数

引数	型	説明
drv	uint16_t	ファイルを開くドライブ番号
did	uint16_t	値 0（ルートディレクトリ）を設定しなければなりません。
fid	uint16_t	R_tfs_alloci 関数から取得されたファイル番号
flags	int16_t	以下の値をフラグに指定することができます。 TFS_O_RDONLY - 読み取り専用として開く TFS_O_WRONLY - 書き込み専用として開く TFS_O_RDWR - 読み取り／書き込みとして開く TFS_O_CREAT - 存在しない場合、新しいファイルを作成します。

戻り値

型	説明
int16_t	関数が正常に実行された場合、ファイル記述子を返します。 関数がエラーで終了した場合、戻り値は-1 です。

使用例

```
int16_t fd;
uint16_t file_no;
// Initialization
file_no = R_tfs_alloci（0,0,0）;
fd = R_tfs_openi（0, 0, file_no, TFS_O_RDWR|TFS_O_CREAT）;
```

6.8　R_tfs_close

プロトタイプ

```
int16_t R_tfs_close （int16_t fd）
```

説明

この関数は、ファイル記述子 fd と関連付けられたファイルを閉じます。

引数

引数	型	説明
fd	int16_t	閉じるファイルと関連付けられたファイル記述子

戻り値

型	説明
int16_t	関数が正常に実行された場合、戻り値は 0 です。 関数がエラーで終了した場合、戻り値は-1 です。

使用例

```
int16_t ret_val, fd;
uint16_t file_no;
// Initialization
file_no = R_tfs_alloci（0,0,0）;
fd = R_tfs_openi（0, 0, file_no, TFS_O_RDWR|TFS_O_CREAT）;
ret_val = R_tfs_close（fd）;
```

6.9　R_tfs_write

プロトタイプ

```
int16_t R_tfs_write （int16_t fd, const void *buf, uint32_t count）
```

説明

この関数は、バッファ buf からファイル記述子 fd と関連付けられたファイルに count バイトを書き込みます。

引数

引数	型	説明
fd	int16_t	データを書き込むファイルと関連付けられたファイル記述子
buf	const void*	書き込むデータを含むバッファを指すポインタ
count	uint32_t	書き込むデータのバイト数

戻り値

型	説明
int16_t	関数が正常に実行された場合、書き込まれた実際のバイト数を返します。 関数がエラーで終了した場合、戻り値は-1 です。

使用例

```
int16_t ret_val, fd;
uint16_t file_no;
// Initialization
fd = R_tfs_openi（0, 0, file_no, TFS_O_RDWR|TFS_O_CREAT）;
ret_val = R_tfs_write（fd,"123456789",9）;
R_tfs_close（fd）;
```

6.10　R_tfs_read

プロトタイプ

```
int16_t R_tfs_read （int16_t fd, void *buf, uint32_t count）
```

説明

この関数は、ファイル記述子 fd と関連付けられたファイルからバッファ buf にデータの count バイトを読み取ります。

引数

引数	型	説明
fd	int16_t	データを読み取るファイルと関連付けられたファイル記述子
buf	void*	書き込むデータを含むバッファを指すポインタ
count	uint32_t	書き込むデータのバイト数

戻り値

型	説明
int16_t	関数が正常に実行された場合、書き込まれた実際のバイト数を返します。 関数がエラーで終了した場合、戻り値は-1 です。

使用例

```
int16_t ret_val, fd;
uint16_t file_no;
// Initialization and other processing
fd = R_tfs_openi（0, 0, file_no, TFS_O_RDWR）;
ret_val = R_tfs_read（fd,rw_buff,9）;
```

6.11　R_tfs_lseek

プロトタイプ

```
int16_t R_tfs_lseek （int16_t fd, int32_t offset, int16_t whence）
```

説明

この関数は、whence で指定された位置からオフセットバイト数だけファイル記述子 fd と関連付けられたファイルポインタを移動します。引数 whence は以下の値を取ります。

Whence 値	ファイルポインタ位置
TFS_SEEK_SET	ファイルの先頭
TFS_SEEK_CUR	現在のファイルポインタ位置
TFS_SEEK_END	ファイルの終わり

引数

引数	型	説明
fd	int16_t	ファイルと関連付けられたファイル記述子
offset	int32_t	ファイルポインタを移動するバイト数
whence	int16_t	ファイルポインタの移動元の位置

戻り値

型	説明
int16_t	関数が正常に実行された場合、ファイルポインタ位置を返します。 関数がエラーで終了した場合、戻り値は-1 です。

使用例

```
int16_t fd;
uint16_t file_no;
int32_t fp;
// Initialization and other processing
fd = R_tfs_openi（0, 0, file_no, TFS_O_RDWR|TFS_O_CREAT）;
fp = R_tfs_lseek（fd, 5,TFS_SEEK_SET）;
```

6.12　R_tfs_removei

プロトタイプ

```
int16_t R_tfs_removei （uint16_t drv, uint16_t did, uint16_t fid）
```

説明

この関数は、ドライブ drv からファイル fid を削除します。ディレクトリ番号 did には値 0（ルートディレクトリ）を設定する必要があります。

引数

引数	型	説明
drv	uint16_t	ファイルの削除元のドライブ番号
did	uint16_t	値 0（ルートディレクトリ）を設定しなければなりません。
fid	uint16_t	削除するファイルのファイル番号

戻り値

型	説明
int16_t	関数が正常に実行された場合、戻り値は 0 です。 関数がエラーで終了した場合、戻り値は-1 です。

使用例

```
int16_t ret_val;
uint16_t file_no;
// Initialization and other processing
ret_val = R_tfs_removei（0,0,file_no）;
```

6.13　R_tfs_stati

プロトタイプ

```
int16_t R_tfs_stati（uint16_t drv, uint16_t did, uint16_t fid, struct tfs_stat *buf）
```

説明

この関数は、ファイル fid のファイル情報を取得し、R_tfs_stat 構造 buf に格納します。

引数

引数	型	説明
drv	uint16_t	ファイルのドライブ番号
did	uint16_t	値 0（ルートディレクトリ）を設定しなければなりません。
fid	uint16_t	情報を取得するファイル
buf	struct tfs_stat*	ファイル情報から構成される関数から受け取った戻り値

戻り値

型	説明
int16_t	関数が正常に実行された場合、戻り値は 0 です。 関数がエラーで終了した場合、戻り値は-1 です。

使用例

```
uint16_t file_no;
struct tfs_stat stat;
int16_t ret_val;
// Initialization and other processing
ret_val = R_tfs_stati（0,0,file_no,&stat）;
```

6.14　R_tfs_statfs

プロトタイプ

```
int16_t R_tfs_statfs （uint16_t drv, struct R_tfs_statfs *buf）
```

説明

この関数は、マウントされたボリュームに関するスペース使用可能性情報を取得します。

引数

引数	型	説明
drv	uint16_t	ボリュームをマウントするドライブ
buf	struct tfs_statfs*	ボリューム情報から構成される関数から受け取った戻り値

戻り値

型	説明
int16_t	関数が正常に実行された場合、戻り値は 0 です。 関数がエラーで終了した場合、戻り値は-1 です。

使用例

```
int16_t ret_val;
struct R_tfs_statfs statfs;
// Initialization and other processing
ret_val = R_tfs_statfs（0,&statfs）;
```

6.15　R_tfs_get_errno

プロトタイプ

```
int16_t R_tfs_get_errno （void）
```

説明

この関数は、直前のライブラリ関数に対応するエラー番号を返します。直前のライブラリ関数が正常に実行された場合、0 が返されます。

引数

なし

戻り値

型	説明
int16_t	TFS ライブラリエラー番号（セクション 5 に記載）

使用例

```
int16_t err_code, fd;
// Initialization and other processing
R_tfs_write（fd,"12345678912345678912345678 9",27）;
err_code = R_tfs_get_errno（）;   //Returns error code corresponding to
                                  //R_tfs_write
```

6.16　R_tfs_get_date

プロトタイプ

```
uint16_t R_tfs_get_date （void）
```

説明

これはユーザ定義関数です。ライブラリはこの関数の定義を含みません。作業環境に基づいてこの関数を実装する必要があります。セクション 4.7 に記載したフォーマットで現在の日付を返すように実装する必要があります。

引数

なし

戻り値

型	説明
uint16_t	セクション 4.7 に記載するフォーマットの現在の日付

使用例

R_tfs_get_date 関数の実装例については、サンプルソフトウェアを参照してください。

6.17　R_tfs_get_time

プロトタイプ

```
uint16_t R_tfs_get_time （void）
```

説明

これはユーザ定義関数です。ライブラリはこの関数の定義を含みません。作業環境に基づいてこの関数を実装する必要があります。セクション 4.7 に記載したフォーマットで現在の時刻を返すように実装する必要があります。

引数

なし

戻り値

型	説明
uint16_t	セクション 4.7 に記載するフォーマットの現在の時刻

使用例

R_tfs_get_time 関数の実装例については、サンプルソフトウェアを参照してください。

7. メモリドライバインタフェース

このセクションでは、メモリドライバインタフェース関数について詳細に説明します。これらの関数のプロトタイプと各関数の実装で必要な処理について説明します。これらの関数の実装は、ユーザに使用可能なメモリドライバとともに使用できるように作成する必要があります。

7.1 関数

TFS によって使用されるドライブは、単一ボリューム（DOS パーティション）互換です。パーティションテーブル情報は TFS から隠されるので、パーティションテーブルを使用する必要がある場合は、ドライバがパーティションテーブルを処理しなければなりません。TFS ライブラリは、ドライブを 64 バイト固定長論理セクタアレイとして使用し、これらの論理セクタ内で I/O を要求します。

7.1.1 R_tfs_write_lsec

プロトタイプ

```
int16_t R_tfs_write_lsec （uint16_t drv, uint32_t lsec, const void *buf）
```

説明

この関数は、ディスクドライブにデータを書き込むためのコードで構成する必要があります。書き込むデータに関する詳細は引数で指定します。この関数は、バッファ buf からドライブ drv の lsec で指定されたボリューム（DOS パーティション適合）論理セクタにデータを書き込みます。

引数

引数	型	説明
drv	uint16_t	ボリュームをマウントするドライブ
lsec	uint32_t	論理セクタ番号を指定します。
buf	const void*	書き込むデータを指すポインタ

戻り値

型	説明
int16_t	関数が正常に実行された場合、戻り値は 0 です。 関数がエラーで終了した場合、戻り値は-1 です。

7.1.2　R_tfs_read_lsec

プロトタイプ

```
int16_t R_tfs_read_lsec （uint16_t drv, uint32_t lsec, void *buf）
```

説明

この関数は、ディスクドライブから読み取るコードで構成する必要があります。読み取るデータに関する詳細は、引数で指定します。この関数は、ドライブ drv の lsec で指定されたボリューム（DOS パーティション適合）論理セクタからバッファ buf にデータを読み取ります。

引数

引数	型	説明
drv	uint16_t	ボリュームをマウントするドライブ
lsec	uint32_t	論理セクタ番号を指定します。
buf	void*	読み取りデータを格納するバッファを指すポインタ

戻り値

型	説明
int16_t	関数が正常に実行された場合、戻り値は 0 です。 関数がエラーで終了した場合、戻り値は-1 です。

8. サンプルプログラム

このセクションでは、Tiny FS ライブラリを使用するためのサンプルプログラムについて説明します。サンプルプログラムは HEW（High-Performance Embedded Workshop）ワークスペースの形式です。ご使用のシステムに合わせてマイクロコンピュータと周辺機器を初期化し修正してください。

8.1 概要

サンプルプログラムはテキストファイルを作成し、データをファイルに書き込み、ファイルに実際に書き込まれたデータを確認します。

プログラムが実行されると、Tinyファイルシステムボリュームが外部メモリカードにマウントされます。メモリカードは外部アドオンボード*[1]を介してRSK*[2]に接続されます。ファイルがメモリカード上に作成され、2KBのテキストデータがファイルに書き込まれます。その後、ファイルは閉じられます。書き込まれたデータを確認するために、ファイルを読み取りモードで再び開きます。ファイルの内容全体が読み取られ、プログラムの書き込みバッファデータと比較されます。データの内容が一致しているかどうかはRSKボード上のLEDに表示されます。

データはヘッダファイル data_file.h に定義されます。

【注】 *1 外部アドオンボードにはメモリメディアを挿入するためのスロットがあります。メモリメディアの端子は RSK の該当する端子に接続されます。この回路はユーザが購入を希望する Renesas Solutions Kits には付属しておらず、ルネサスから入手することはできません。
*2 RSK は RX610 の Renesas Starter Kit を略したものです。

8.2　フロー

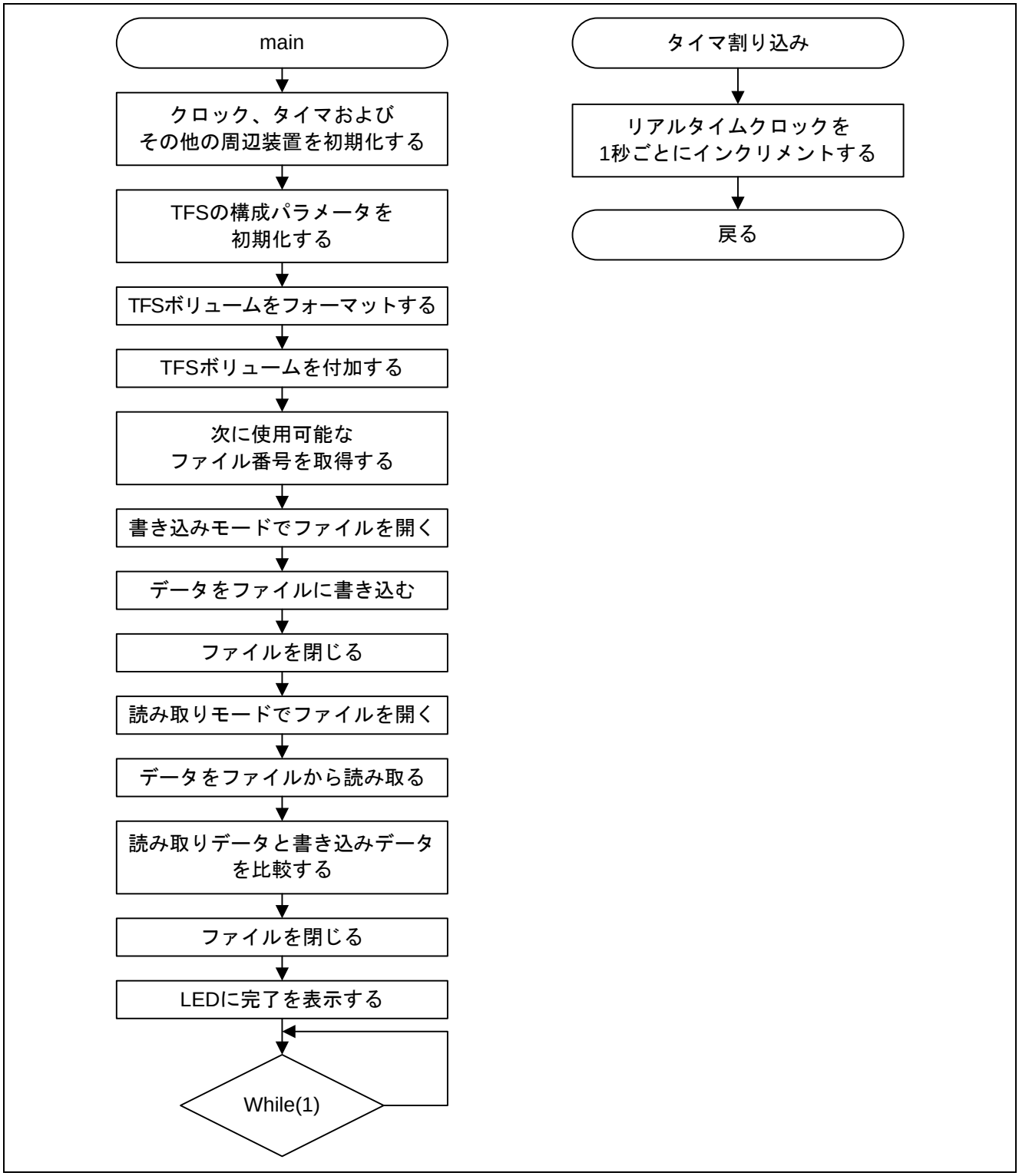

図 1　サンプルプログラムのフロー

8.3　関数リスト

以下の表にサンプルプログラムにある関数のリストを示します。

No.	関数名	概要
1.0	main	ファイルにデータを書き込み、書き込まれたデータを読み取り、確認します。
1.1	R_init_clock	マイクロコンピュータおよびその他のクロックと関連するレジスタのクロックが初期化されます。
1.2	R_init_portpins	周辺装置のポート端子を初期化します。
1.3	R_init_1sTimer	タイマはリアルタイムクロック実装のためにセットアップされます。
1.4	R_error	エラー処理関数
1.5	R_mmc_drv_init	メモリドライバ初期化
1.6	R_tfs_init	ライブラリ構成を初期化します - ライブラリ関数
1.7	R_tfs_format1	メモリカードをフォーマットします - ライブラリ関数
1.8	R_tfs_attach	ドライブを TFS ボリュームにマウントします - ライブラリ関数
1.9	R_tfs_alloci	次に使用可能なファイル番号を取得します - ライブラリ関数
1.10	R_tfs_openi	ファイルを開きます - ライブラリ関数
1.11	R_tfs_write	データをファイルに書き込みます - ライブラリ関数
1.12	R_tfs_read	データをファイルから読み取ります - ライブラリ関数
1.13	R_tfs_close	ファイルを閉じます - ライブラリ関数
1.14	R_tfs_detach	ドライブをアンマウントします - ライブラリ関数
1.15	R_tfs_exit	ライブラリの終了処理 - ライブラリ関数
2.0	R_int_timer_CMI0A	リアルタイムクロックを毎秒インクリメントします。

8.4　関数チャート

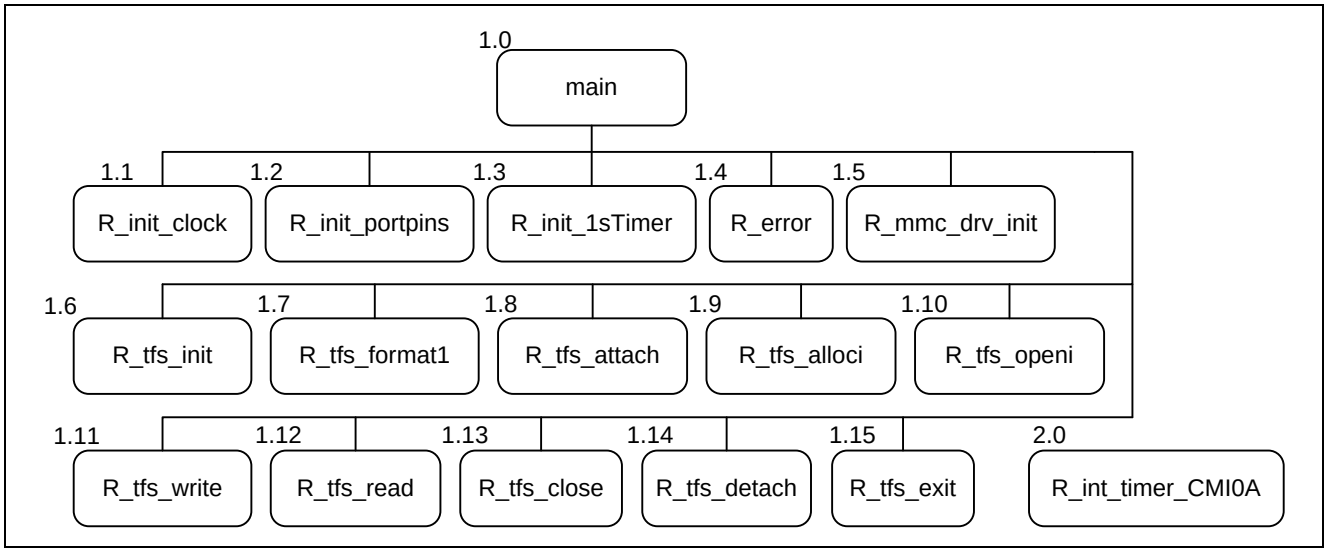

図 2　関数チャート

8.5　ワークスペースにおけるフォルダ作成

```
tfs_sample_RX600                          ワークスペースディレクトリ
|
|
|--  R5F56108                              プロジェクトディレクトリ
       |
       |--   Debug                         構成ディレクトリ
       |
       |--   Debug_RX600_E1_E20_SYSTEM     構成ディレクトリ
       |
       |--   lib                           TFS ファイルシステムライブラリ格納ディレクトリ
       |
       |--   Release                       構成ディレクトリ
       |
       |--   sample                        サンプルソース格納ディレクトリ
       |
       |--   hew_files                     HEW 自動生成ファイル格納ディレクトリ
```

9. サンプルソフトウェアの使用方法

このセクションでは、サンプルソフトウェアの実行の詳細について説明します。

9.1 サンプルソフトウェアの実行

- サンプルソフトウェアワークスペースを作成して、abs ファイルを RSK にダウンロードします。
- "Reset Go"ボタンをクリックすると、プログラムが実行されます。
- 最初にファイル書き込み動作が実行されます。新しいテキストファイルがメモリカードに作成され、2KB のテキストデータが書き込まれます。その後、ファイルは閉じられます。
- 同じファイルが読み取りモードで再び開かれます。ファイルの内容が読み取られ、ファイルを書き込むときに渡されたデータと比較されます。これは、書き込み関数によりファイルに書きこまれたデータが実際に正しくファイルに書きこまれたかどうかを確認するために行われます。
- プログラムの現在の状態は RSK ボード上の LED に表示されます。
- 以下の表はプログラム実行に対応する LED 表示を示します。

LED0	LED1	意味
ON	OFF	プログラムが実行中です。
ON	ON	実行が成功しました。
OFF	ON	エラーが発生しました。

9.2 リアルタイムクロック

サンプルソフトウェアには、タイマを使用したリアルタイムクロック実装を含んでいます。タイマは 1 秒ごとに割り込みを生成するように構成されています。対応する割り込みサービスルーチンで、現在の時刻と日付がインクリメントされます。この時刻と日付は、一部のファイル操作に使用されます。時刻と日付の格納に関する詳細については、セクション 4.7 を参照してください。

9.3 ファイル読み取り／書き込みのためのサンプルデータ

ファイル読み取り／書き込みのためのサンプルデータは、ヘッダファイル data_file.h に格納されます。データは 2048 個の要素のアレイに格納され、合計サイズは 2KB（2048 バイト）です。データアレイは繰り返し書き込まれる"Renesas"テキスト文字列から構成されます。必要に応じて、このアレイおよび対応するマクロ FILESIZE を変更することができます。

10. ライブラリの特性

このセクションでは、ライブラリのメモリ使用量についての詳細を説明します。

10.1　占有メモリサイズ

マイクロコンピュータ	ROM	RAM
RX600	5649	158

単位：バイト

10.2　占有スタックサイズ

関数	RX600
R_tfs_init	16
R_tfs_exit	8
R_tfs_format1	156
R_tfs_attach	60
R_tfs_detach	16
R_tfs_alloci	52
R_tfs_openi	88
R_tfclose	32
R_tfs_write	128
R_tfs_read	124
R_tfs_lseek	4
R_tfs_removei	72
R_tfs_stati	40
R_tfs_statfs	60
R_tfs_get_errno	4

単位：バイト

10.3　ファイルシステムデータ構造によって占有されるメモリ

構造	1 つの構造変数のメモリ
	RX600
tfs_volume	28
tfs_file	16
tfs_buff	72
tfs_config	12
tfs_format_param1	16
tfs_stat	12
tfs_statfs	10

単位：バイト

上の表を使用して、ユーザのアプリケーションの異なる TFS ライブラリ構造変数に必要なメモリを計算することができます。1 つの構造変数に必要なメモリに変数の数を掛けると、その特定の構造のすべての変数に必要なメモリが得られます。

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	2010.10.08	—	初版発行

製品ご使用上の注意事項

ここでは、マイコン製品全体に適用する「使用上の注意事項」について説明します。個別の使用上の注意事項については、本ドキュメントおよびテクニカルアップデートを参照してください。

1. 未使用端子の処理
   【注意】未使用端子は、本文の「未使用端子の処理」に従って処理してください。
   CMOS 製品の入力端子のインピーダンスは、一般に、ハイインピーダンスとなっています。未使用端子を開放状態で動作させると、誘導現象により、LSI 周辺のノイズが印加され、LSI 内部で貫通電流が流れたり、入力信号と認識されて誤動作を起こす恐れがあります。未使用端子は、本文「未使用端子の処理」で説明する指示に従い処理してください。
2. 電源投入時の処置
   【注意】電源投入時は，製品の状態は不定です。
   電源投入時には、LSI の内部回路の状態は不確定であり、レジスタの設定や各端子の状態は不定です。
   外部リセット端子でリセットする製品の場合、電源投入からリセットが有効になるまでの期間、端子の状態は保証できません。
   同様に、内蔵パワーオンリセット機能を使用してリセットする製品の場合、電源投入からリセットのかかる一定電圧に達するまでの期間、端子の状態は保証できません。
3. リザーブアドレス（予約領域）のアクセス禁止
   【注意】リザーブアドレス（予約領域）のアクセスを禁止します。
   アドレス領域には、将来の機能拡張用に割り付けられているリザーブアドレス（予約領域）があります。これらのアドレスをアクセスしたときの動作については、保証できませんので、アクセスしないようにしてください。
4. クロックについて
   【注意】リセット時は、クロックが安定した後、リセットを解除してください。
   プログラム実行中のクロック切り替え時は、切り替え先クロックが安定した後に切り替えてください。
   リセット時、外部発振子（または外部発振回路）を用いたクロックで動作を開始するシステムでは、クロックが十分安定した後、リセットを解除してください。また、プログラムの途中で外部発振子（または外部発振回路）を用いたクロックに切り替える場合は、切り替え先のクロックが十分安定してから切り替えてください。
5. 製品間の相違について
   【注意】型名の異なる製品に変更する場合は、製品型名ごとにシステム評価試験を実施してください。
   同じグループのマイコンでも型名が違うと、内部 ROM、レイアウトパターンの相違などにより、電気的特性の範囲で、特性値、動作マージン、ノイズ耐量、ノイズ輻射量などが異なる場合があります。型名が違う製品に変更する場合は、個々の製品ごとにシステム評価試験を実施してください。

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

Colophon 1.0