LED/SW Board
ソフトウェアマニュアル

Version 1.0.4

SIL00-U00
SIL00-U01

株式会社アットマークテクノ
http://www.atmark-techno.com/

SUZAKU 公式サイト
http://suzaku.atmark-techno.com/

目次

表目次

図目次

1. はじめに

1.1. マニュアルについて

このマニュアルは、SUZAKU I/O シリーズのLED/SWボード（SIL00-U01*）に付属している以下のサンプルソフトウェアについて記載されています。

- アプリケーションプログラム
- Linux 用デバイスドライバ

ソフトウェアのカスタマイズなどの開発作業を行う際には、参考文献 [1][2][3]もあわせてご覧ください。SUZAKU の機能を最大限に引き出すために、ご活用いただければ幸いです。

1.2. フォントについて

このマニュアルでは以下のようにフォントを使っています。

表 1-1 使用しているフォント

フォント例	説明
本文中のフォント	本文
`[PC ~]$ ls`	プロンプトとユーザ入力文字列 ソースファイルのコード、ファイル名、ディレクトリ名など

1.3. コマンド入力例の表記について

このマニュアルに記載されているコマンドの入力例は、表示されているプロンプトによって、それぞれに対応した実行環境を想定して書かれています。「/」の部分はカレントディレクトリによって異なります。各ユーザのホームディレクトリは「~」で表わします。

表 1-2 表示プロンプトと実行環境の関係

プロンプト	コマンドの実行環境
`[PC /]#`	作業用 PC 上の特権ユーザで実行
`[PC /]$`	作業用 PC 上の一般ユーザで実行
`[SUZAKU /]#`	SUZAKU 上の特権ユーザで実行
`[SUZAKU /]$`	SUZAKU 上の一般ユーザで実行

1.4. 数字の表記について

このマニュアルに記載されている数字は、特に明記されている場合を除き表記方法によって異なった基数を表します。

* SIL00-U00 をお使いの方は製品型番を読み替えてご利用ください。動作における違いはございません。

表 1-3 表示方法と基数の関係

ベース	記載方法	備考
2 進数	10100101b	数字の後に”b”と記載
10 進数	165	0 から 9 までのアラビア数字
16 進数	0xA5	“0x”を数字の前に記載

1.5. 謝辞

SUZKAU で使用しているソフトウェアは Free Software / Open Source Software で構成されています。Free Software / Open Source Software は世界中の多くの開発者の成果によって成り立っています。この場を借りて感謝の意を示します。

1.6. ソフトウェアに関する注意事項

本製品に含まれるソフトウェア(付属のドキュメント等も含みます)は、現状のまま(AS IS)提供されるものであり、特定の目的に適合することや、その信頼性、正確性を保証するものではありません。また、本製品の使用による結果についてもなんら保証するものではありません。

1.7. 保証に関する注意事項

- 製品保証範囲について

付属品（ソフトウェアを含みます）を使用し、取扱説明書、各注意事項に基づく正常なご使用に限り有効です。万一正常なご使用のもと製品が故障した場合は、初期不良保証期間内であれば新品交換をさせていただきます。

- 保証対象外になる場合

次のような場合の故障・損傷は、保証期間内であっても保証対象外になります。

1. 取扱説明書に記載されている使用方法、または注意に反したお取り扱いによる場合
2. 改造や部品交換に起因する場合。または正規のものではない機器を接続したことによる場合
3. お客様のお手元に届いた後の輸送、移動時の落下など、お取り扱いの不備による場合
4. 火災、地震、水害、落雷、その他の天災、公害や異常電圧による場合
5. AC アダプタ、専用ケーブルなどの付属品について、同梱のものを使用していない場合
6. 修理依頼の際に購入時の付属品がすべて揃っていない場合

- 免責事項

弊社に故意または重大な過失があった場合を除き、製品の使用および、故障、修理によって発生するいかなる損害についても、弊社は一切の責任を負わないものとします。

本製品は購入時の初期不良以外の保証をおこなっておりません。保証期間は商品到着後２週間です。本製品をご購入されましたらお手数でも必ず動作確認をおこなってからご使用ください。本製品に対して注意事項を守らずに発生した故障につきましては保証対象外となります。

2. 作業の前に

2.1. 準備するもの

SIL00-U01 を使用する前に、以下のものを準備してください。

- SUZAKU
  SUZAKU-S シリーズまたは SUZAKU-V シリーズのいずれかの SUZAKU ボードです。

- 作業用 PC
  Linux もしくは Windows が動作し、1 ポート以上のシリアルポートを持つ PC です。

- シリアルクロスケーブル
  D-Sub9 ピン（メス－メス）の「クロス接続用」ケーブルです。

- D-Sub 9 ピン-10 ピン変換ケーブル
  D-Sub9 ピンと SUZAKU のピンヘッダ(10 ピン)を接続するケーブルです。

- 付属 CD-ROM（以降、付属 CD）
  SUZAKU に関する各種マニュアルやソースコードが収納されています。

- シリアル通信ソフトウェア
  minicom や Tera Term などのシリアル通信ソフトウェアです。作業用 PC にインストールしてください。

- 電源
  AC アダプタ 5V を使用してください。

2.2. 接続方法

下の図を参照して、シリアルクロスケーブルと LAN ケーブルを SUZAKU に、そして電源（AC アダプタ）を LED/SW ボードに接続してください。

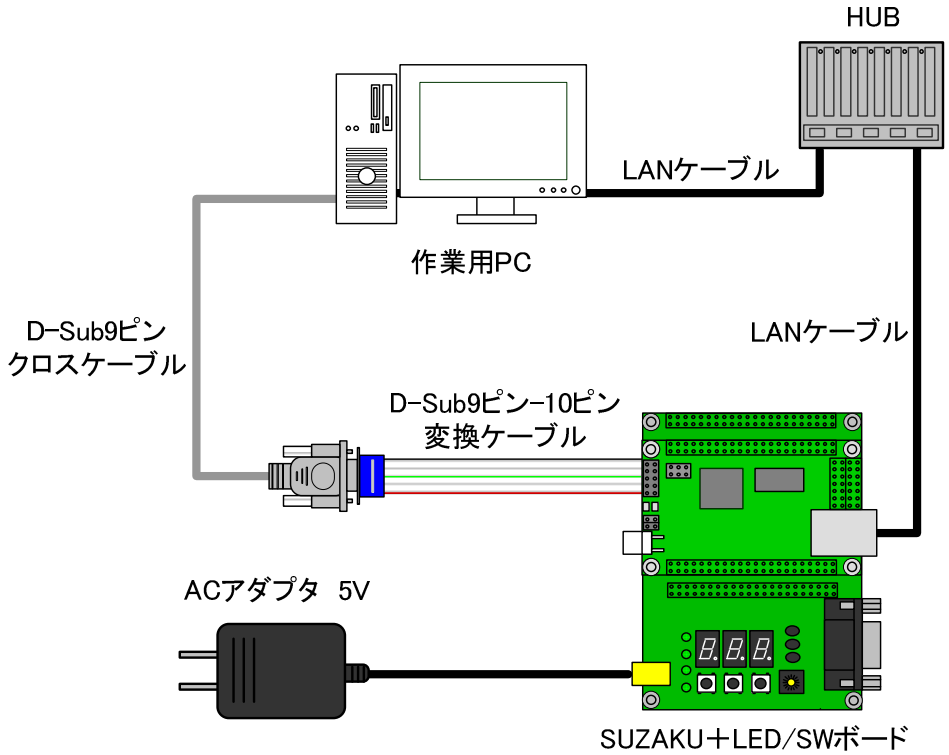

図 2-1 SIL00-U01 接続例

注意

SUZAKU ボードからは絶対に電源を供給しないでください。電源がショートし、機器を破損する可能性があります。SUZAKU ボードへの電源は、LED/SW ボードから供給されるようになっています。

2.3. 開発環境の構築について

SUZAKUでのクロス開発環境の構築については、参考文献 [3][5]を参照してください。

2.3.1. FPGA データ

SUZAKU-SまたはSUZAKU-Vスターターキットとして SIL00-U01 を購入された場合、SUZAKUのフラッシュメモリに書き込まれているFPGAのデータは 参考文献 [4]用になっています。このため本書で説明しているアプリケーションおよびデバイスドライバを使う前に、フラッシュメモリ内の FPGAコンフィギュレーションデータを変更する必要があります。変更する FPGA データは、付属 CD の `suzaku-starter-kit/image`ディレクトリに、`fpga-sz`*`XXX`*`-sil-gpio_control.bin` という名のファイルです。`sz`*`XXX`*の部分は、お使いのSUZAKUボードの型番です。

FPGAのデータは以下のようにhermitコマンドで書き換えることが可能です。hermitの詳細については、参考文献 [3]をご覧ください。

```
[PC ~]$ hermit download -r fpga -i fpga-szXXX-sil-gpio_control.bin
--force-locked
```

図 2-2 FPGA データの書き換え

注意

FPGA リージョンを不正なデータで書き換えたり、書き換えが異常終了した場合は SUZAKU および SIL00-U01 の電源を入れないでください。最悪の場合、SUZAKU および SIL00-U01 を破壊する恐れがあります。書き換えに失敗した場合は、お使いの SUZAKU 用ハードウェアマニュアルを参照し正しいデータに書き戻してから電源を入れてください。

2.4. インターフェースの表記について

SIL00-U01 の各種インターフェースの配置と、本マニュアルでの表記は以下のとおりです。

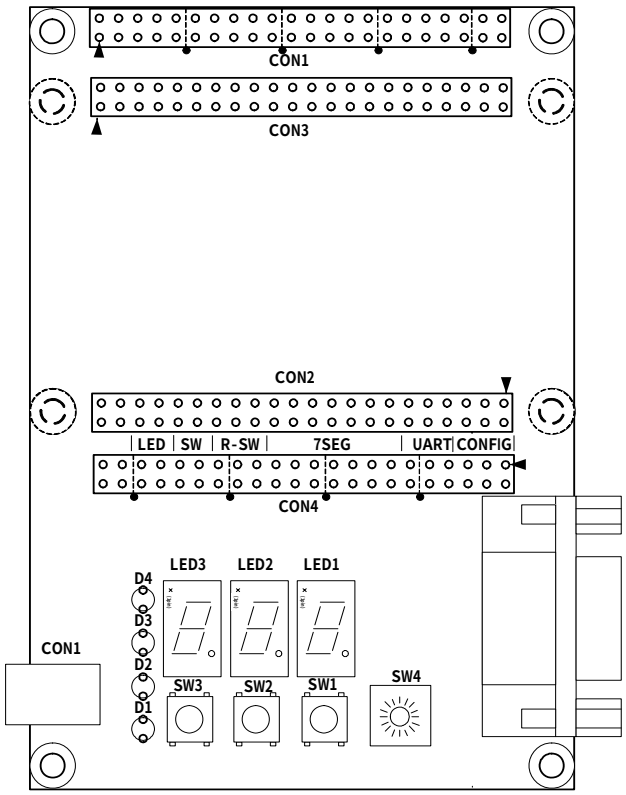

図 2-3 各種インターフェースの配置

表 2-1 各種インターフェースの表記

インターフェース	表記
単色 LED	D1, D2, D3, D4
7 セグメント LED	LED1, LED2, LED3
押しボタンスイッチ	SW1, SW2, SW3
ロータリコードスイッチ	SW4

また、7 セグメント LED の各セグメントの位置と表記を下図に示します。

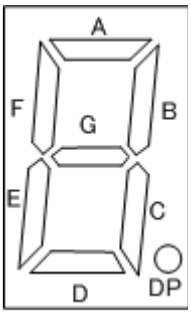

図 2-4 7 セグメント LED のセグメント

3. アプリケーションプログラム

この章では SIL00-U01 用に用意されているアプリケーションプログラムを説明します。すべてのアプリケーションは C 言語で記述されています。またソースコード自体の見通しが良くなるように、それぞれ単機能に特化した作りになっています。

SIL00-U01 に搭載している各インターフェースによって、以下のようなアプリケーションプログラムが用意されています。

- 単色 LED: `demo-led`
- 7 セグメント LED: `demo-7seg`
- 押しボタンスイッチ: `demo-sw`
- ロータリコードスイッチ: `demo-rsw`

なお、これらのアプリケーションプログラムは、次章で説明するデバイスドライバを使用します。実行する際には、まず、デバイスドライバ（「4.6. ドライバの選択」参照）およびアプリケーション（「3.6. アプリケーションの選択」参照）を選択後、ビルドし、イメージファイルを作成します。その後、フラッシュメモリの書き換えを行なってください。

3.1. 単色 LED

3.1.1. 概要

この章では単色 LED を制御するアプリケーションプログラムについて説明します。単色 LED は SIL00-U01 の左側に一列に並んでいます。この単色 LED を操作するアプリケーションプログラムの名前は `demo-led` です。`demo-led` は `/bin` にインストールされています。このアプリケーションプログラムは後述するオプションの組み合わせによって任意の単色 LED の操作や、状態を表示することが可能です。

たとえば `demo-led` に引数として `0` を与えることで、すべての単色 LED を消灯することができます。逆に引数として `f` を与えることで、すべての単色 LED を点灯させることができます。

3.1.2. 単色 LED の状態表現

`demo-led`では SIL00-U01 に搭載されている 4 つの単色LEDの状態を数字で表します。各単色LEDの点灯を 1、消灯を 0 とするバイナリ表現に置き換えると、表 3-1のように 4 bitですべての単色LEDの状態を表すことができます。

上記の例のように 0x0（0000b）はすべての単色 LED が消灯している状態を、0xf（1111b）ですべての単色 LED が点灯している状態を表わします。また、D1 と D3 のみ点灯される状態は 0x5（0101b）と表現します。

表 3-1 単色 LED のビットマップ

3	2	1	0
↑	↑	↑	↑
D4	D3	D2	D1

3.1.3. 使用法

demo-led の使用法について説明します。

表 3-2 demo-led の使用法

書式	demo-led [options] [ctrl]
説明	プログラム demo-led は、単色 LED を制御するアプリケーションプログラムです。引数 ctrl は制御コードで、単色 LED の状態表現で指定します。オプションの指定により任意の LED 毎の制御が行えます。ctrl の指定は任意で、未指定時は、全 LED の状態を出力します。
オプション	-l N　対象の LED を指定します。N は、1（D1）～4（D4）の数字を指定します。未指定時は、全 LED が対象です。 -u　使用法を表示します。 -v　バージョンを表示します。

```
# demo-led 0          (全 LED を消灯)
# demo-led f          (全 LED を点灯)
# demo-led            (全 LED の状態を表示)
f
# demo-led -l 2 0     (LED2 を消灯)
# demo-led -l 4 1     (LED4 を点灯)
# demo-led -l 4       (LED4 の状態を表示)
1
```

図 3-1 demo-led の実行例

3.2. 7 セグメント LED

3.2.1. 概要

SIL00-U01 の中央に位置する 3 つの 7 セグメント LED を制御するアプリケーションプログラムについて説明します。このアプリケーションプログラムの名前は demo-7seg です。demo-7seg は他のアプリケーションプログラム同様 /bin にインストールされています。

demo-7seg を引数無しで実行することで、すべての 7 セグメント LED の状態を表示することができます。

3.2.2. 7 セグメント LED の状態表現

demo-7segでは SIL00-U01 に搭載されている 7 セグメントLEDの状態を数字で表します。各 7 セグメントLEDには名前が示す通り 7 つのセグメントLEDが搭載されており、これらのセグメントには 図 2-4が示す通り AからGの名前が与えられています。また、小数点を表す 8 つ目のセグメントは DPと呼ばれます。

これら 8 つのセグメントを demo-7seg では下表のようにバイナリで表現しています。1 がセグメント LED の点灯を、0 がセグメント LED の消灯を表します。

表 3-3 セグメントのビットマップ（7 セグメント LED）

7	6	5	4	3	2	1	0
↑	↑	↑	↑	↑	↑	↑	↑
DP	G	F	E	D	C	B	A

例えば 2 という数字を表現する場合を考えます。2 は、A, B, G, E, D の 5 つのセグメント LED の点灯で表現することができます。その際の状態は 0x5b（0101 1011b）と表します。

上記のように一つの 7 セグメント LED を表すには 8bit 必要になります。SIL00-U01 に搭載されている 7 セグメント LED は 3 つありますので、すべての 7 セグメント LED を表現するために 24 bit 使用します。

表 3-4 7 セグメント LED のビットマップ

31～24	23～16	15～8	7～0
↑	↑	↑	↑
(空)	LED3	LED2	LED1

3.2.3. 使用法

`demo-7seg` の使用法について説明します。

表 3-5 demo-7seg の使用法

書式	`demo-7seg [options] [ctrl]`	
説明	プログラム`demo-7seg`は、7 セグメントLEDを制御するアプリケーションプログラムです。引数 `ctrl` は制御コードで、LEDの状態を 16 進数で指定します（「3.2.2. 7 セグメントLEDの状態表現」を参照）。オプションの指定により任意のLEDやセグメント毎の制御も行えます。	
オプション	`-l N`	制御する LED の番号を指定します。`N` には 1（LED1）～3（LED3）を指定できます。未指定時は、全 LED が対象です。
	`-u`	使用法を表示します。
	`-v`	バージョンを表示します。

```
# demo-7seg              (全 LED の状態出力)
4f666d
# demo-7seg 4f66fd       (LED3: 「3」点灯, LED2:「4」点灯, LED1:「5」点灯)
# demo-7seg -l 1 6d      (LED1: 「5」点灯)
```

図 3-2 demo-7seg の実行例

3.3. 押しボタンスイッチ

3.3.1. 概要

押しボタンスイッチの押下状態を確認するサンプルアプリケーション demo-sw について説明します。
demo-sw は SIL00-U01 の手前側に 3 つ並んでいる押しボタンスイッチの状態を標準出力に表示します。
後述するオプションを使用することで、指定したスイッチの状態のみ表示することも可能です。

3.3.2. 使用法

demo-sw の使用法について説明します。

表 3-6 demo-sw の使用法

書式	demo-sw [options]	
説明	プログラム demo-sw は、押しボタンスイッチの押下状態を出力するアプリケーションプログラムです。オプションによりボタン毎の状態を出力することができます。	
オプション	-l N	監視するスイッチの番号を指定します。N には 1（SW1）～3（SW3）を指定できます。未指定時は、全スイッチが対象です。
	-u	使用法を表示します。
	-v	バージョンを表示します。

```
# demo-sw
SW3:[ ] SW2:[ ] SW1:[ ]
SW3:[ ] SW2:[ ] SW1:[ ]
SW3:[ ] SW2:[*] SW1:[ ]      ← SW2 が押された
  :
```

図 3-3 demo-sw の実行例

demo-sw を停止する場合は、Ctrl + C キーで実行を中断してください。

3.4. ロータリコードスイッチ

3.4.1. 概要

ロータリコードスイッチの状態を表示するサンプルアプリケーション demo-rsw について説明します。
ロータリコードスイッチはツマミを回すことで状態を変化させるスイッチのことで、SIL00-U01 は 押しボタンスイッチの右側に 16 の状態を表現できるロータリコードスイッチを搭載しています。
demo-rsw は 16 の状態を一定間隔で標準出力に表示します。

3.4.2. 使用法

demo-rsw の使用法について説明します。

表 3-7 demo-rsw の使用法

書式	demo-rsw [options]
説明	プログラム demo-rsw は、ロータリコードスイッチの状態を表示するアプリケーションプログラムです。スイッチが示す位置を標準出力に書き出します。
オプション	-u 使用法を表示します。 -v バージョンを表示します。

```
# demo-rsw
0
0
4
:
```

図 3-4 demo-rsw の実行例

demo-rsw を停止する場合は、Ctrl + C キーで実行を中断してください。

3.5. シリアルポート

3.5.1. 概要

シリアルポート用の特別なアプリケーションは用意されておりません。既存の UNIX コマンド等で利用する例を説明します。

SIL00-U01 のシリアルポートは、一般的なシリアルボートと同じですので、一般的な Linux のシリアルポート用アプリケーションプログラムが動作します。ここでは、tip と呼ばれるシリアル通信ツールを使って説明します。

3.5.2. tip

tipは、シンプルなシリアル通信プログラムです。ここではそれぞれのシリアルポートに別のシリアル通信ソフトウェアが接続されていると仮定します。tipを起動するには使用するシリアルポートや転送レートなどいくつかの通信設定が必要です。指定値については、「表 4-20 シリアル通信設定」を参照してください。

以下に、シリアルポートに ttyS1、転送レート 115200bps を指定した、tip コマンドの実行例を示します。

```
[SUZAKU ~/]# tip -s 115200 -l /dev/ttyS1
```

図 3-5 tip コマンドの実行例

tip コマンドの終了は、「~.」と入力します。

3.6. アプリケーションの選択

本章で紹介した各種アプリケーションは、全て atmark-dist および uClinux-dist に含まれています。

- `atmark-dist/vendors/AtmarkTechno/SUZAKU-V.Common/sil`
- `uClinux-dist/user/suzaku/sil`

ご利用の際には、`make menuconfig` で、アプリケーションを追加する必要があります。

```
[PC ~/atmark-dist]$ make menuconfig
Main Menu
  Kernel/Library/Defaults Selection   --->
    [*] Customize Vendor/User Settings

Userland Configuration
  Vendor specific
    --- SUZAKU I/O LED/SW Board Sample Application
    [*] demo-led
    [ ] demo-7seg
    [ ] demo-sw
    [ ] demo-rsw

  Miscellaneous Applications   --->
     :
    [*] tip
     :
```

図 3-6 アプリケーションの選択例（atmark-dist）

```
[PC ~/uClinux-dist]$ make menuconfig
Main Menu
  Kernel/Library/Defaults Selection   --->
    [*] Customize Vendor/User Settings

Main Menu
  Miscellaneous Applications   --->
     :
    [*] tip
     :
    --- SUZAKU I/O LED/SW Board Sample Application
    [*] demo-led
    [ ] demo-7seg
    [ ] demo-sw
    [ ] demo-rsw
```

図 3-7 アプリケーションの選択例（uClinux-dist）

4. デバイスドライバ

この章では SIL00-U01 に実装されている各種インターフェースを制御するための Linux 用デバイスドライバについて説明します。

4.1. 単色 LED

4.1.1. 概要

SIL00-U01 には、単色 LED（緑）が 4 個実装されています。ここで説明するデバイスドライバは、4 個の単色 LED を個別または全て同時に、制御（点灯、消灯）することができます。

表 4-1 単色 LED デバイスドライバ

ドライバ名	sil-led
ドライバ説明	SUZAKU I/O LED/SW Board LED Driver
デバイスファイル名	/dev/silled （全部）
	/dev/silled1 （D1）
	/dev/silled2 （D2）
	/dev/silled3 （D3）
	/dev/silled4 （D4）
ソースファイル所在	▼atmark-dist の場合 linux-2.6.x/drivers/char/sil-led.c linux-2.6.x/include/asm-ppc/suzaku_sil.h ▼uClinux-dist の場合 linux-2.4.x/drivers/char/sil-led.c linux-2.4.x/include/asm-microblaze/suzaku_sil.h

4.1.2. システムコール

本ドライバで用意されているシステムコールは、open・close・read・write の 4 種類です。それぞれについて説明します。

open

表 4-2 open システムコール（単色 LED）

書式	int open(const char *pathname, int flags);
説明	デバイスをオープンします。オープンに成功した場合、新しいファイルディスクリプタを返します。
引数	pathname　オープンするデバイスファイル名 flags　ファイルアクセスモード O_RDONLY, O_WRONLY, O_RDWR のどれかひとつを指定します。それぞれ読み込み専用、書き込み専用、読み書き用にファイルをオープンすることを要求します。
返り値	成功した場合は新しいファイルディスクリプタを返し、エラーが発生した場合は-1 を返します。

close

表 4-3 close システムコール（単色 LED）

書式	`int close(int fd);`
説明	デバイスをクローズします。
引数	`fd` ファイルディスクリプタ
返り値	成功した場合は 0 を返し、エラーが発生した場合は-1 を返します。

read

表 4-4 read システムコール（単色 LED）

書式	`ssizte_t read(int fd, void *buf, size_t count);`
説明	デバイスからデータを読み込みます。最大 `count` バイトをバッファ `buf` へ読み込みます。 読み込みデータは、readシステムコールを呼び出した時点の単色LEDの状態を示します。単色LEDの状態については、「3.1.2. 単色LEDの状態表現」を参照してください。
引数	`fd` ファイルディスクリプタ `buf` 読み込みデータを格納するバッファ `count` 読み込みデータのバイト数
返り値	成功した場合は読み込んだバイト数を返し、エラーが発生した場合は-1 を返します。

write

表 4-5 write システムコール（単色 LED）

書式	`int write(int fd, const void *buf, size_t count);`
説明	デバイスへデータを書き込みます。バッファ `buf` から最大 `count` バイト分のデータをデバイスへ書き込みます。 書き込みデータには、制御したい単色LEDの状態を示す文字を指定します。単色LEDの状態は、「3.1.2. 単色LEDの状態表現」を参照してください。
引数	`fd` ファイルディスクリプタ `buf` 書き出しデータを格納するバッファ `count` 書き出しデータのバイト数
返り値	成功した場合は書き込んだバイト数を返し、エラーが発生した場合は-1 を返します。

4.2. 7 セグメント LED

4.2.1. 概要

SIL00-U01 には、7 セグメント LED が 3 個実装されています。ここで説明するドライバは、3 個の 7 セグメント LED を個別にまたは全部同時に、制御（点灯、消灯）することができます。

表 4-6 7 セグメント LED デバイスドライバ

ドライバ名	sil-7seg
ドライバ説明	SUZAKU I/O LED/SW Board 7SegmentLED Driver
デバイスファイル名	`/dev/sil7seg` （全部）
	`/dev/sil7seg1` （LED1）
	`/dev/sil7seg2` （LED2）
	`/dev/sil7seg3` （LED3）

ソースファイル所在	▼`atmark-dist` の場合
	`linux-2.6.x/drivers/char/sil-7seg.c`
	`linux-2.6.x/include/asm-ppc/suzaku_sil.h`
	
	▼`uClinux-dist` の場合
	`linux-2.4.x/drivers/char/sil-7seg.c`
	`linux-2.4.x/include/asm-ppc/suzaku_sil.h`
	`linux-2.4.x/include/asm-microblaze/suzaku_sil.h`

4.2.2. システムコール

本ドライバで用意されているシステムコールは、open・close・read・write の 4 種類です。それぞれについて説明します。

open

表 4-7 open システムコール（7 セグメント LED）

書式	`int open(const char *pathname, int flags);`
説明	デバイスをオープンします。オープンに成功した場合、新しいファイルディスクリプタを返します。
引数	`pathname` オープンするデバイスファイル名 `flags` ファイルアクセスモード `O_RDONLY`, `O_WRONLY`, `O_RDWR` のどれかひとつを指定します。それぞれ読み込み専用、書き込み専用、読み書き用にファイルをオープンすることを要求します。
返り値	成功した場合は新しいファイルディスクリプタを返し、エラーが発生した場合は-1 を返します。

close

表 4-8 close システムコール（7 セグメント LED）

書式	`int close(int fd);`
説明	デバイスをクローズします。
引数	fd　　ファイルディスクリプタ
返り値	成功した場合は 0 を返し、エラーが発生した場合は-1 を返します。

read

表 4-9 read システムコール（7 セグメント LED）

書式	`ssizte_t read(int fd, void *buf, size_t count);`
説明	デバイスからデータを読み込みます。最大 count バイトをバッファ buf へ読み込みます。 読み込みデータは、readシステムコールを呼び出した時点の 7 セグメントLEDの状態を示します。状態については、「3.2.2. 7 セグメントLEDの状態表現」を参照してください。
引数	fd　　ファイルディスクリプタ buf　　読み込みデータを格納するバッファ count　　読み込みデータのバイト数
返り値	成功した場合は読み込んだバイト数を返し、エラーが発生した場合は-1 を返します。

write

表 4-10 write システムコール（7 セグメント LED）

書式	`int write(int fd, const void *buf, size_t count);`
説明	デバイスへデータを書き込みます。バッファ buf から最大 count バイト分のデータをデバイスへ書き込みます。 書き込みデータには、対象の 7 セグメントLEDの制御状態を示す文字を指定します。状態については、「3.2.2. 7 セグメントLEDの状態表現」を参照してください。
引数	fd　　ファイルディスクリプタ buf　　書き出しデータを格納するバッファ count　　書き出しデータのバイト数
返り値	成功した場合は書き込んだバイト数を返し、エラーが発生した場合は-1 を返します。

4.3. 押しボタンスイッチ

4.3.1. 概要

SIL00-U01 には、押しボタンスイッチが 3 個実装されています。ここで説明するドライバは、3 個の押しボタンスイッチの状態を取得することができます。

表 4-11 押しボタンスイッチデバイスドライバ

ドライバ名	sil-sw
ドライバ説明	SUZAKU I/O LED/SW Board Switch Driver
デバイスファイル名	`/dev/silsw` （全部）
	`/dev/silsw1` （SW1）
	`/dev/silsw2` （SW2）
	`/dev/silsw3` （SW3）

ソースファイル所在	▼atmark-dist の場合
	`linux-2.6.x/drivers/char/sil-sw.c`
	`linux-2.6.x/include/asm-ppc/suzaku_sil.h`
	
	▼uClinux-dist の場合
	`linux-2.4.x/drivers/char/sil-sw.c`
	`linux-2.4.x/include/asm-ppc/suzaku_sil.h`
	`linux-2.4.x/include/asm-microblaze/suzaku_sil.h`

4.3.2. 押しボタンスイッチの状態表現

本デバイスドライバでは、押しボタンスイッチの状態を次のように表現します。1 つの押しボタンスイッチの状態として、押下と開放の 2 つを定義し、それぞれを 1 と 0 で表します。

SIL00-U01 には、3 つの押しボタンスイッチが搭載されており、各スイッチの状態を 表 4-12に示す並びで整列させることで、3 ビットの 2 進数とみなすことができます。この 2 進数を 16 進数に変換した値を、全スイッチの状態として使用します。

表 4-12 押ボタンスイッチのビットマップ

2	1	0
↑	↑	↑
SW3	SW2	SW1

例えば、SW1 と SW3 が押下、その他を開放とした全スイッチの状態は、「5」と表現されます。

4.3.3. システムコール

本ドライバで用意されているシステムコールは、open・close・read の 3 種類です。それぞれについて説明します。

open

表 4-13 open システムコール（押しボタンスイッチ）

書式	`int open(const char *pathname, int flags);`
説明	デバイスをオープンします。オープンに成功した場合、新しいファイルディスクリプタを返します。
引数	`pathname` オープンするデバイスファイル名 `flags` ファイルアクセスモード `O_RDONLY` を指定します。読み込み専用でファイルをオープンすることを要求します。
返り値	成功した場合は新しいファイルディスクリプタを返し、エラーが発生した場合は-1 を返します。

close

表 4-14 close システムコール（押しボタンスイッチ）

書式	`int close(int fd);`
説明	デバイスをクローズします。
引数	fd ファイルディスクリプタ
返り値	成功した場合は 0 を返し、エラーが発生した場合は-1 を返します。

read

表 4-15 read システムコール（押しボタンスイッチ）

書式	`ssizte_t read(int fd, void *buf, size_t count);`
説明	デバイスからデータを読み込みます。最大 `count` バイトをバッファ `buf` へ読み込みます。 読み込みデータは、readシステムコールを呼び出した時点の押しボタンスイッチの状態を示します。押しボタンスイッチの状態は、「4.3.2. 押しボタンスイッチの状態表現」を参照してください。
引数	`fd` ファイルディスクリプタ `buf` 読み込みデータを格納するバッファ `count` 読み込みデータのバイト数
返り値	成功した場合は読み込んだバイト数を返し、エラーが発生した場合は-1 を返します。

4.4. ロータリコードスイッチ

4.4.1. 概要

SIL00-U01 には、ロータリコードスイッチが 1 個実装されています。ここで説明するドライバは、ロータリコードスイッチの状態を取得することができます。

表 4-16 ロータリコードスイッチデバイスドライバ

ドライバ名	sil-rsw
ドライバ説明	SUZAKU I/O LED/SW Board Rotary Switch Driver
デバイスファイル名	/dev/silrsw
ソースファイル所在	▼atmark-dist の場合 linux-2.6.x/drivers/char/sil-rsw.c linux-2.6.x/include/asm-ppc/suzaku_sil.h ▼uClinux-dist の場合 linux-2.4.x/drivers/char/sil-rsw.c linux-2.4.x/include/asm-microblaze/suzaku_sil.h

4.4.2. システムコール

本ドライバで用意されているシステムコールは、open・close・read の 3 種類です。

open

表 4-17 open システムコール（ロータリコードスイッチ）

書式	int open(const char *pathname, int flags);
説明	デバイスをオープンします。オープンに成功した場合、新しいファイルディスクリプタを返します。
引数	pathname　オープンするデバイスファイル名 flags　ファイルアクセスモード O_RDONLY を指定します。読み込み専用でファイルをオープンすることを要求します。
返り値	成功した場合は新しいファイルディスクリプタを返し、エラーが発生した場合は-1 を返します。

close

表 4-18 close システムコール（ロータリコードスイッチ）

書式	int close(int fd);
説明	デバイスをクローズします。
引数	fd　ファイルディスクリプタ
返り値	成功した場合は 0 を返し、エラーが発生した場合は-1 を返します。

read

表 4-19 read システムコール（ロータリコードスイッチ）

書式	`ssizte_t read(int fd, void *buf, size_t count);`
説明	デバイスからデータを読み込みます。最大 count バイトをバッファ buf へ読み込みます。 読み込みデータは、read システムコールを呼び出した時点のロータリコードスイッチの状態を示します。ロータリコードスイッチの状態は、0～15 の 16 段階を 16 進数で表した文字として表現されます。例えば、状態 0 は ‘0’、状態 10 は’a’となります。
引数	`fd` ファイルディスクリプタ `buf` 読み込みデータを格納するバッファ `count` 読み込みデータのバイト数
返り値	成功した場合は読み込んだバイト数を返し、エラーが発生した場合は-1 を返します。

4.5. シリアルポート

4.5.1. 概要

uartlite用のデバイスドライバを使用しています。シリアル通信の各種設定値は、表 4-20のようになっています。

表 4-20 シリアル通信設定

項目	設定値
転送レート	115200bps
データ長	8bit
ストップビット	1bit
パリティ	なし
フロー制御	なし
デバイスファイル名	`/dev/ttyS1`

4.6. ドライバの選択

この章で紹介した各種デバイスドライバは、全て付属 CD の atmark-dist または uClinux-dist に含まれています。ご利用の際には、使用したいデバイスドライバを `make menuconfig` で追加し、再度ビルドする必要があります。

```
[PC ~/atmark-dist]$ make menuconfig
Main Menu
  Kernel/Library/Defaults Selection  --->
    [*] Customize Kernel Settings

Linux Kernel Configuration
  Device Drivers  --->
    Character devices  --->
      [*] SUZAKU I/O LED/SW Board
      < >   Led Support
      <*>   7 segment led support
      < >   Switch support
      < >   Rotary code switch support
      [ ]   RS232C support
```

図 4-1 ドライバの選択例 (atmark-dist)

```
[PC ~/uClinux-dist]$ make menuconfig
Main Menu
  Kernel/Library/Defaults Selection   --->
    [*] Customize Kernel Settings

Main Menu
  Character devices   --->
    [*] SUZAKU I/O LED/SW Board
    < >   Led Support
    <*>   7 segment led support
    < >   Switch support
    < >   Rotary code switch support
    [ ]   RS232C support
```

図 4-2 ドライバの選択例 (uClinux-dist)

5. 参考文献

[1] 『atmark-dist Developers Guide』, ㈱アットマークテクノ.
[2] 『uClinux-dist Developers Guide』, ㈱アットマークテクノ.
[3] 『SUZAKU ソフトウェアマニュアル』, ㈱アットマークテクノ.
[4] 『SUZAKU スターターキットガイド（FPGA 開発編）』, ㈱アットマークテクノ.
[5] 『SUZAKU スターターキットガイド（Linux 開発編）』, ㈱アットマークテクノ.
[6] 『OPB General Purpose Input/Output (GPIO) Datasheet』, Xilinx.

改訂履歴

Ver	年月日	改訂内容
1.0.0	2006.08.11	・初版発行
1.0.1	2006.09.15	・誤記訂正
1.0.2	2006.10.20	・「1.7．保証に関する注意事項」を追加
1.0.3	2006.12.15	・表紙デザイン改版 ・uClinux-dist-20051110-suzaku6 用に書き換え
1.0.4	2007.10.10	・SIL00-U01 に対応 ・atmark-dist 用の記述を追加

LED/SW Board ソフトウェアマニュアル　　2007 年 10 月 10 日　Version 1.0.4

株式会社アットマークテクノ
060-0035 札幌市中央区北 5 条東 2 丁目 AFT ビル 6F　　TEL:011-207-6550　FAX:011-207-6570