

I2CMonitor ソフトウェア説明書

概要

「I2CMonitor」は、弊社 SV シリーズ基板のカスタム機能である I2C スレーブ機能を Linux 環境からコントロールするサンプルソフトウェアです。本ソフトウェアは SV0 ボードで受信された I2C データのリアルタイムのモニタリングが可能であり、データはコンソールにテキストとして随時出力されます。ソフトウェアはコマンドラインベースとなっており、スクリプトや外部アプリケーションからボードの操作が可能になっています。

ソフトウェアのビルドには「UVC SDK」内の「libsvm」ライブラリ（Linux 環境より弊社 SV シリーズのコマンド送受信を扱うためのライブラリ）のインストールが必要です。「libsvm」ライブラリについては弊社「UVC SDK」のマニュアルを参照してください。

動作環境

- OS: Ubuntu (16.04 LTS 64bit にて動作確認)
- USB ポート: USB3.0 または USB2.0 搭載
- 対応ボード: SV0-03 (カスタムコード "OHT0")

ファイル説明

I2CMonitorIF.cpp / I2CMonitorIF.h

FPGA 実装の I2C スレーブの初期化や、ボードからデータを取得しテキスト形式にフォーマットする関数を実装したクラス (CI2CMonitorIF) です。アプリケーションよりこのクラスを継承して使用します。FPGA レジスタアクセス関数 ReadFPGARegister / WriteFPGARegister などは仮想関数となっているので、継承先のクラスで実装する必要があります。

CI2CMonitorIF クラス

関数
<pre>void DecodeI2CByte(int d, std::string& text);</pre> <p>– d: FPGA から読み出した I2C バッファの値を指定します。</p> <p>– text: フォーマットされたテキストを返します。</p> <p>FPGA の I2C バッファの生の値 (1 ワード) から、テキスト形式にフォーマットします。</p>
<pre>void FormatI2CMonitorData(const unsigned short* data, int cnt, int wordMode, std::string& text);</pre> <p>– data: FPGA の I2C バッファから読み出した配列を指定します。</p>

- cnt: data のサイズを指定します。
- wordMode: (0: 8bit, 1: 16bit)
- text: フォーマットされたテキストを返します。

FPGA の I2C バッファから読み出した生の配列から、パケット (Start Condition - Stop Condition まで) あたり 1 行のテキスト形式にフォーマットします。FormatI2CMonitorData() ではステータスビットは無視して、アドレスおよびデータをカンマ区切りのテキストにフォーマットします。

```
int InitI2CSlave(int deviceAddr);
```

- deviceAddr: 応答すべきデバイスアドレスを指定
- 返回值: 0 で正常

FPGA の I2C スレーブ機能を初期化します。

```
int PollingExample(std::string& text, int decodeType, int wordMode = 0);
```

- text: テキストにフォーマットした I2C バッファを返します。
- decodeType: (0: Simple, 1: complex)
- wordMode: (0: 8bit, 1: 16bit)
- 返回值: I2C 受信が 1 パケット以上行われた場合、行われた時点の FPGA バッファの残りバイト数を返す

ポーリング関数の実装例です。FPGA の I2C バッファを読み出し、テキスト形式にフォーマットして text に返します。I2C バッファが空の場合すぐに返回值 0 を返します。

decodeType 引数によりフォーマット形式を 2 種類から選択します。

```
virtual UINT FPGARegisterIF_Init();
```

- 返回值: 0 で正常

FPGA レジスタアクセスに初期化処理が必要な場合、この関数を実装してください。

PollingExample() の最初に呼び出されます。

```
virtual UINT FPGARegisterIF_End();
```

- 返回值: 0 で正常

FPGA レジスタアクセスに終了処理が必要な場合、この関数を実装してください。

PollingExample() の最後に呼び出されます。

```
virtual UINT ReadFPGARegister(UINT regAddr, UINT* pData, int cnt);
```

- regAddr: レジスタアドレス
- pData: 受信する値へのポインタ
- cnt: 受信するレジスタ数

FPGA のレジスタ読み込みを行う関数です。継承先クラスで実装してください。 実装例は I2CMonitorIF_L.h を参照してください。
<pre>virtual UINT WriteFPGARegister(UINT regAddr, UINT data);</pre> <ul style="list-style-type: none">- regAddr: レジスタアドレス- data: 送信する値へのポインタ
FPGA のレジスタ書き込みを行う関数です。継承先クラスで実装してください。

I2CMonitorIF_L.h

libsvm ライブラリを使用した ReadFPGARegister / WriteFPGARegister 関数の実装です。

I2CMonitor.cpp

I2CMonitor サンプルのメインとなるファイルです。libsvm をインストールした状態で、gcc でコンパイルすることができます。I2C スレーブ機能を実装してプログラムを作成する際は、I2CMonitor.cpp と I2CMonitorIF.cpp を参考に実装してください。

本サンプルの動作は、プログラムの引数をパースした後、メインループ内で一定周期ごとに
CI2CMonitorIF::PollingExample() 関数を呼び出しています。PollingExample() でデータが受信され、
テキストデータが返された場合、このテキストデータをコンソールに出力します。プログラムの終了は
Ctrl+C にて行ってください。

コマンド説明

I2CMonitor [option(s)]

オプション一覧

option	value
-format [n]	コンソールに表示するテキストのフォーマットを指定します。 [n] には 0 または 1 を指定します。 0: Simple (コマンド単位にフォーマット) 1: Complex (イベント単位にフォーマット) 指定しない場合、Simple となります。
-interval [n]	ポーリング周期を ms 単位で指定します。 指定しない場合、100ms となります。

-addr [x]	応答すべきデバイスアドレスを 16 進数で指定します。 7F を指定すると、すべてのデバイスアドレス (0x00-0x7F) に応答します。 指定しない場合、7F となります。
-wordmode [n]	-wordmode オプションは -format 0 が指定されたとき有効で、I2C データをテキストにフォーマットする際のレジスタのビット幅を指定します。 [n] は 0 または 1 を指定します。 0: 8bit 1: 16bit 指定しない場合、0 が設定されます。
-device [n]	0 から始まるデバイス番号により、開くデバイスを指定します。 ここで指定するデバイス番号は OS にマウントされた Video Capture デバイスの番号を指定します。SVMctl で指定したボード ID とは異なります。

実行例

外部より送信した I2C データ

- (1) デバイスアドレス 0x40, レジスタ 0x20 に 4 バイト書き込み {0x01, ..., 0x04}
- (2) デバイスアドレス 0x40, レジスタ 0x20 から 4 バイト読み出し {0xFF, ..., 0xFF}

-format 0 を指定した場合

```
./I2CMonitor -format 0 -interval 33 -addr 7F
```

```
a@dynabook:~/Desktop/SDK_Work/I2CMonitor$ ./I2CMonitor -format 0 -interval 33 -a
ddr 7F
press Ctrl-C to exit.
W,40,20,01,02,03,04,
R,40,20,FF,FF,FF,FF,
^C
```

-format 0 を指定した場合、1 行 1 パケットとして、下記フォーマットに従って出力されます。

(1), (2), (3), (4)
(1): {W, R} W: 書き込みアクセスを示す

R: 読み込みアクセスを示す
(2): デバイスアドレス (7bit)
(3): レジスタアドレス (8bit もしくは 16bit)
(4): データ (カンマ区切り)

-format 1 を指定した場合

```
./I2CMonitor -format 1 -interval 33 -addr 7F
```

```
a@dynabook:~/Desktop/SDK_Work/I2CMonitor$ ./I2CMonitor -format 1 -interval 33 -a
ddr 7F
press Ctrl-C to exit.
[180]S, [140]A:40, [1E0]W, [1F0]As, [020]D:20, [1F0]As, [001]D:01, [1F0]As, [002
]D:02, [1F0]As, [003]D:03, [1F0]As, [004]D:04, [1F0]As, [1C0]P,
[1F8], [180]S, [140]A:40, [1E0]W, [1F0]As, [020]D:20, [1F0]As, [1A0]Sr, [140]A:4
0, [1E8]R, [1F0]As, [0FF]D:FF, [1F4]Am, [0FF]D:FF, [1F4]Am, [0FF]D:FF, [1F4]Am,
[0FF]D:FF, [1F6]Nm, [1C0]P,
^C
```

-format 1 を指定した場合、イベント単位のデータとして、1 パケット 1 行のテキストが出力されま
す。

[(1)] (2), ...

(1): I2C バッファの生データ (9bit)
(2): デコードされたテキスト
"D:xx": データ
"A:xx": デバイスアドレス
"S": スタートビット
"Sr": リスタートビット
"P": ストップビット
"W": デバイスアドレスがライトを示す
"R": デバイスアドレスがリードを示す
"As": スレーブが ACK を返した
"Ns": スレーブが NACK を返した
"Am": マスタが ACK を返した
"Nm": マスタが NACK を返した

注意

ポーリング周期が遅い場合、FPGA のバッファオーバーフローにより受信データが失われることがあります。FPGA のデータバッファは 1024 ワードですが、I2C 1 パケット (Start Bit から Stop Bit まで) 当たりおよそ 11-16 ワード程度必要なので、パケット当りに換算すると FPGA に実装されているデータバッファ長は 100 パケット弱となります。バッファが一杯になるまでにポーリング関数を呼び出す必要があるため、100 パケット弱のデータが受信されるより早くポーリングを行う必要があるという計算になります。ただし、ポーリング関数内で実行している FPGA のレジスタアクセスの処理時間などにより、ポーリングの最短周期には制約があります。本サンプルソフトの実装では 1 秒あたり約 300 コマンドが正常に受信できる目安となります。

改版履歴

[19/10/15] 初版 (V1.0) 作成