

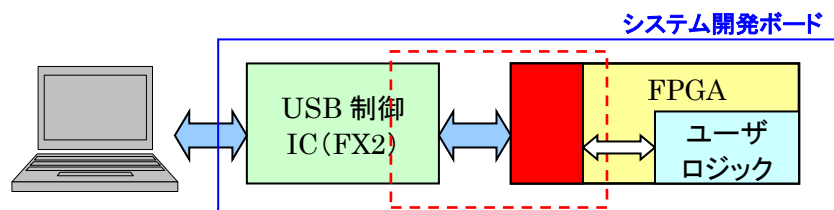
Intel 社ツール“Platform Designer”を利用した Smart-USB Plus 製品用リファレンス回路

GPIF-AVALON ブリッジ回路

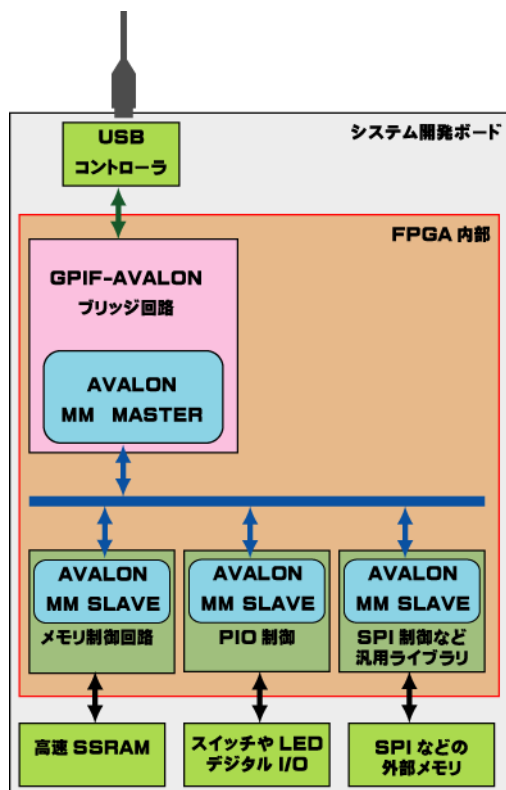
1. QuartusPrime の Platform Designer システム統合ツールで利用できる “GPIF-AVALON ブリッジ”とは？

GPIF-AVALON ブリッジとは、当社製 USB2.0 システムコアである「Smart-USB Plus 製品」の外部 USB インターフェース「GPIF」と、Intel 社製 FPGA 専用内部接続バス「AVALON」を相互に接続する為のバスブリッジです。

(ここで示す GPIF とは、USB 制御 IC (以下、FX2) と FPGA 間の接続のこと (図 1 で赤表示した部分) です。)



<図 1. システムブロック図>



<図 2. GPIF-AVALON ブリッジのブロック図>

従来、Platform Designer では、AVALON バス・マスタとして Nios2 プロセッサが必須でしたが、GPIF-AVALON ブリッジを使用することで、Nios2 の代わりに AVALON バス・ペリフェラルを利用することができます。例えば、SPI や I2C 通信など Nios2 を実装せずに、ホスト PC から USB インターフェースにより AVALON バスに接続した SPI ライブラリを直接アクセスすることができます。この様に、製品添付の制御ソフトウェア RefApp7 又はお客様が開発した制御アプリケーションを使用し、ツールに用意される無償の回路ライブラリに、ホスト PC から USB 経由で直接アクセスすることが可能です。

また、GPIF-AVALON ブリッジは Nios2 とも共存できるので、PC と Nios2 間でデータのやりとりが可能です。このため、USB インタフェース付きのマイコンボードとして運用ができます。

2. GPIF-AVALON ブリッジの目的

- ✓ Smart-USB Plus 製品ファミリの使いやすさを向上
- ✓ 各種 Smart-USB Plus 製品内でのマイグレーション性を向上
- ✓ 豊富な無償 IP の有効利用
- ✓ Nios2 がなくても、USB 付きのマイコンボード化を実現

3. 使用環境

GPIF-AVALON ブリッジ (GPIF_Master) は、以下の環境での動作を確認しています。ただし、Platform Designer 用コンポーネント全ての動作を保証する訳ではありません。動作確認をしているコンポーネントは限定的ですので注意して下さい。

対応ツールバージョン： Quartus Prime 17.0 以降 (Lite Edition でも動作します)

※Nios2 を使用しない限り、Nios2 開発ツールをインストールする必要はありません。

サンプル FPGA プロジェクト (Card-UNIV6.qar) は、CX-Card10 システム開発ボード用です。オプションボードの Card-UNIV6 と組み合わせて利用できます。以下 URL から無償ダウンロードできます。

http://www.prime-sys.co.jp/Download/GPIF_Avalon/Card-UNIV6.zip

サンプルプロジェクトは Smart-USB Plus 製品ファミリで Intel FPGA 搭載製品なら、すべてのボードに適用することができますが、各ボード製品で FPGA のピン設定を実施してください。

[GPIF_Master](#) の最新版は Ver1.7 です。

4. GPIF-AVALON ブリッジの利用方法

< CX-Card10 での具体例 >

ボード構成：CX-Card10 とオプションボードの Card-UNIV6 の組み合わせ。

サンプル回路： Card-UNIV6.zip

機能：Card-UNIV6 搭載のセンサ (I2C/SPI) 制御、および PIO により LCD 表示する内容です。

解凍後、Card-UNIV6.qsr ファイルと GPIF_Master1_7 フォルダができます。FPGA サンプル回路は、Quartus Prime 20.1 Lite edition (以下、QP) のプロジェクトを qsr 形式に圧縮しています。QP のツリー

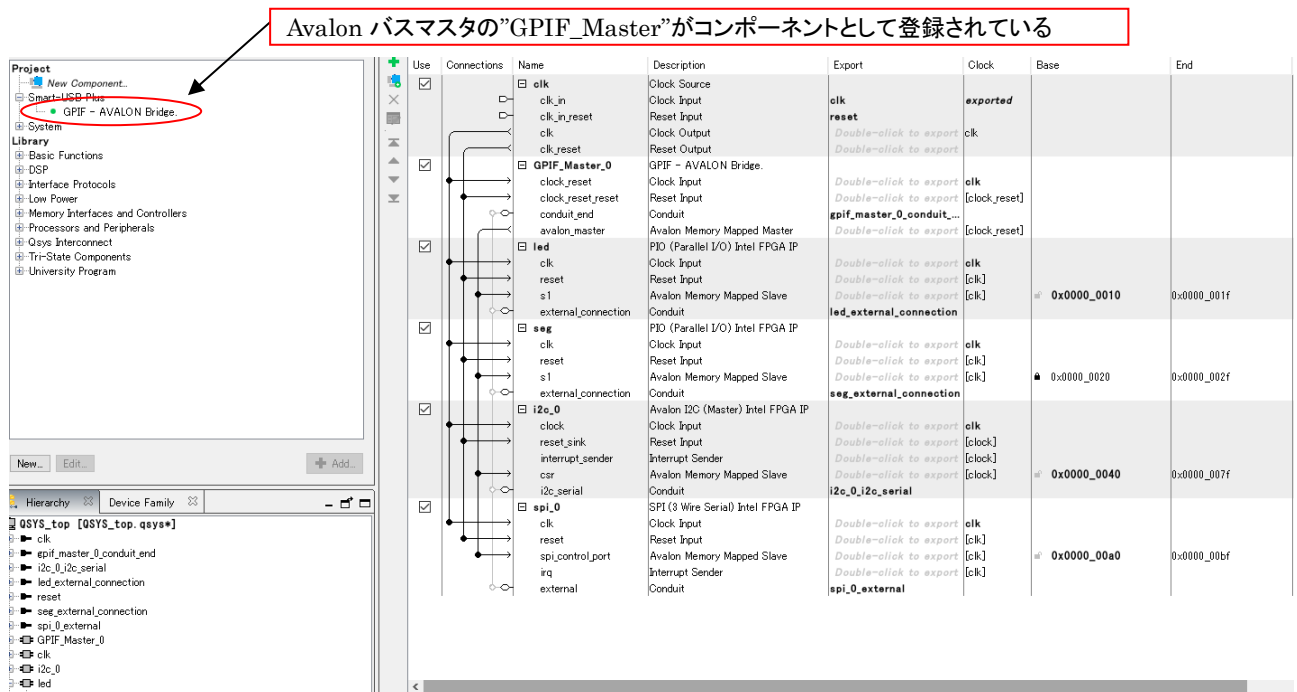
ルバー「Project→Restore Archived Project...」から、qsr を選択して解凍します。その後、プロジェクトフォルダ下にある¥CY10 フォルダに GPIF_Master1_7 フォルダを移動してください。

QP で “Start Analysis & Synthesis”を実行すると、Project Navigator 欄にデザインファイルの階層構成が表示されます。PD の起動は、QP のツールバー“Tools”から“Platform Designer”を選択します。次に PD プロジェクトを選択する画面になるので、ここで“QSYS_top.qsys”を選択し、PD を起動します。

※ここでエラーが発生した場合、GPIF_Master1_7 フォルダがプロジェクトのルートディレクトリにありません。qsr 解凍後にできた¥CY10 フォルダに正しく移動してください。

図 3 に示す System Contents 画面では、“Component Library”欄に示されるコンポーネントを追加することで、Avalon バスペリフェラルを追加できます。

Avalon バスマスタの“GPIF_Master”がコンポーネントとして登録されている



Export	Clock	Base	End
clk	exported		
reset	clk		
Double-click to export			
Double-click to export			
clk	clk		
Double-click to export			
Double-click to export			
gpiif_master_0_conduit...	clk		
Double-click to export			
Double-click to export			
clk	clk	0x0000_0010	0x0000_001f
Double-click to export			
Double-click to export			
led	clk		
Double-click to export			
Double-click to export			
led_external_connection	clk		
Double-click to export			
Double-click to export			
clk	clk	0x0000_0020	0x0000_002f
Double-click to export			
Double-click to export			
seg	clk		
Double-click to export			
Double-click to export			
seg_external_connection	clk		
Double-click to export			
Double-click to export			
clk	clk	0x0000_0040	0x0000_007f
Double-click to export			
Double-click to export			
i2c_0	clk		
Double-click to export			
Double-click to export			
clk	clk	0x0000_00a0	0x0000_00bf
Double-click to export			
Double-click to export			
spi_0	clk		
Double-click to export			
Double-click to export			
spi_0_external	clk		
Double-click to export			

<図 3. PD オープニング画面>

【Avalon バスペリフェラルの追加】

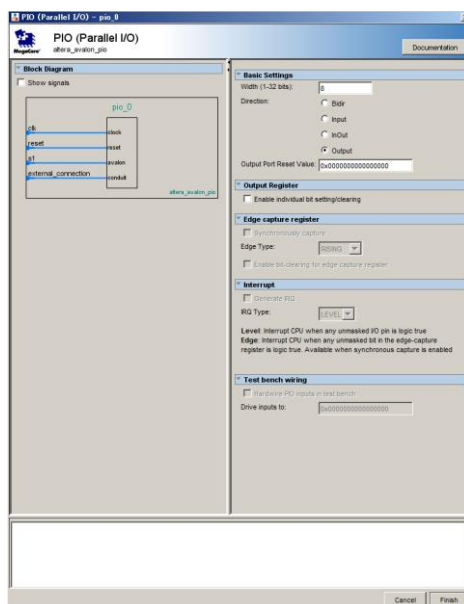
図 3 に示すシステムに、PIO コンポーネントを利用し、ハードウェア・レジスタを追加する場合の例を示します。

サンプル回路では、Avalon バスマスタである“GPIF_Master_0” コンポーネントのベースアドレスを“0x00000000” に設定しています。各 PIO ペリフェラルのベースアドレスは、以下の表 1 の通りです。

コンポーネント名	ベースアドレス (Hex)	備考	RefApp7 のレジスタ操作タブから 制御できるレジスタ番号(Dec)
led (LED)	0x00000010	16bit 幅設定 (出力)	4
seg (7 セグメント)	0x00000020	32bit 幅設定 (出力)	8
i2c_0 (I2C マスタ)	0x00000040		16
spi_0 (SPI マスタ)	0x000000A0	16bit 3 wire, 2MHz	40

<表 1. メモリマップ>

このサンプル回路に 16bit 幅のレジスタを 1 個（出力専用）追加し、Card-UNIV6 の LCD 制御をします。
PD “Component Library”欄の Library→Processors and Peripherals→Peripherals→PIO (Parallel I/O)
を選択し Add ボタンを押してください。



<図 4. PIO 設定画面>

PIO コンポーネントの設定画面は左図の通りです。
Basic Settings 欄で、設定したいレジスタビット幅を指定します。
この場合は 16 です。 LCD 制御は出力だけなので Output を選択
します。

The screenshot displays the Prime Systems PD tool interface. The IP Catalog on the left lists various components, with 'PIO (Parallel I/O) Intel FPGA IP' selected. The Connections panel in the center shows a list of components and their connections. The Messages panel at the bottom shows two error messages:

Type	Path	Message
Error	QSYS_top.pio_0.pio_0.clk	QSYS_top.pio_0.pio_0.clk must be connected to a clock output
Error	QSYS_top.pio_0.pio_0.reset	QSYS_top.pio_0.pio_0.reset must be connected to a reset source

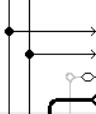
<図 5. PIO を 1 個追加した初期画面>

図 5 では GPIF_Master との接続やクロック、リセット系配線を行っていません。各 PIO コンポーネントの“Clock Input”、“Reset Input”、“Avalon Memory Mapped Slave” 3 カ所のポートにある、Connection 欄の白丸をクリックして黒丸にします。これで追加した PIO が Avalon バスに接続されたことになります。

次に、追加した PIO の名称を lcd に変更し、external_connection ポートの Export 欄をダブルクリックします。この信号が Avalon バスシステムと外部回路を接続するための信号名称になります。この例では、自動的に lcd_external_connection_export という信号名になります。この時点では Avalon バスシステムが完成していないので、PD ツール上にはエラー表示が発生しています。

最後に、追加した PIO のベースアドレスを “0x00000080” (PIO_0 出力専用) に設定します。

※LCD 制御には、コマンド/データ種別信号 RS、イネーブル信号 E、データ信号 DB[7:0] の 10bit だけ使
用します。QP トップファイルの配線では、lcd_external_connection [9:0]だけ FPGA ピンに接続します。

System: QSYS_top Path: lcds1							
Use	Connections	Name	Description	Export	Clock	Base	End
<input checked="" type="checkbox"/>		clk clk_in clk_in_reset clk clk_reset	Clock Source Clock Input Reset Input Clock Output Reset Output	clk reset Double-click to export Double-click to export	exported clk		
<input checked="" type="checkbox"/>		GPIF_Master_0 clock_reset clock_reset_reset conduit_end avalon_master	GPIF - AVALON Bridge. Clock Input Reset Input Conduit Avalon Memory Mapped Master	Double-click to export Double-click to export gpiif_master_0_conduit_... Double-click to export	clk [clock_reset] [clock_reset]		
<input checked="" type="checkbox"/>		led clk reset s1 external_connection	PIO (Parallel I/O) Intel FPGA IP Clock Input Reset Input Avalon Memory Mapped Slave Conduit	Double-click to export Double-click to export Double-click to export led_external_connection	clk [clk] [clk]	0x0000_0010	0x0000_001f
<input checked="" type="checkbox"/>		seg clk reset s1 external_connection	PIO (Parallel I/O) Intel FPGA IP Clock Input Reset Input Avalon Memory Mapped Slave Conduit	Double-click to export Double-click to export Double-click to export seg_external_connection	clk [clk] [clk]	0x0000_0020	0x0000_002f
<input checked="" type="checkbox"/>		i2c_0 clock reset_sink interrupt_sender csr i2c_serial	Avalon I2C (Master) Intel FPGA IP Clock Input Reset Input Interrupt Sender Avalon Memory Mapped Slave Conduit	Double-click to export Double-click to export Double-click to export Double-click to export i2c_0_i2c_serial	clk [clock] [clock] [clock]	0x0000_0040	0x0000_007f
<input checked="" type="checkbox"/>		lcd clk reset	PIO (Parallel I/O) Intel FPGA IP Clock Input Reset Input	Double-click to export Double-click to export	clk [clk]	0x0000_0080	0x0000_008f
<input checked="" type="checkbox"/>		spi_0 external_connection clk reset spi_control_port irq external	Conduit SPI (3 Wire Serial) Intel FPGA IP Clock Input Reset Input Avalon Memory Mapped Slave Interrupt Sender Conduit	Double-click to export Double-click to export Double-click to export Double-click to export spi_0_external	clk [clk] [clk] [clk]	0x0000_00a0	0x0000_00bf

<図 6. PD 設定作業の完了>

コンポーネント名	ベースアドレス (Hex)	備考	RefApp7 のレジスタ操作タブから 制御できるレジスタ番号(Dec)
led (LED)	0x00000010	16bit 幅設定 (出力)	4
seg (7 セグメント)	0x00000020	32bit 幅設定 (出力)	8
i2c_0 (I2C マスタ)	0x00000040		16
lcd (LCD 表示)	0x00000080	16bit (出力)	32
spi_0 (SPI マスタ)	0x000000A0	16bit 3 wire, 2MHz	40

<表 2. PIO を追加して完成したシステムの最終アドレスマップ>

【Avalon システムの生成】

PD の System Contents 画面でシステム構成が完了したら、Generation 画面に移動し、出力ファイルのフォルダを確認後、「Generate」ボタンをクリックしてください。エラーがなければ PD での作業は完了です。

以上で、PD を利用した Avalon システム設計が完了し、Avalon バスモジュールが完成しました。Avalon バスモジュールは、QP プロジェクトフォルダの ¥QSYS_top¥sysntehis¥QSYS_top.v です。

【Avalon システムモジュールをインスタンスエイト】

PD ツールバーの「Generate→Show Instantiation Template...」を選択し、その内容をトップモジュールの Card-UNIV6_lcd.v に、記述してください。追加した PIO ポート分を修正します。

```
QSYS_top u0 (  
    .clk_clk                (pll_clk),  
    .led_external_connection_export (led_wire),  
    .reset_reset_n         (rstn),  
    .seg_external_connection_export (seg_wire),  
    .gpif_master_0_conduit_end_fd  (fd),  
    .gpif_master_0_conduit_end_ctl ({ 1'b0, 1'b0, rgdtn, cmdn, wrn, rdn } ),  
    .gpif_master_0_conduit_end_rdy (rdy_wire),  
    .lcd_external_connection      (lcd_wire),  
    .i2c_0_i2c_serial_sda_in      (sda_wire),  
    .i2c_0_i2c_serial_scl_in      (scl_wire),  
    .i2c_0_i2c_serial_sda_oe      (sda_oe),  
    .i2c_0_i2c_serial_scl_oe      (scl_oe),  
    .spi_0_external_MISO          (bme_sdo),  
    .spi_0_external_MOSI          (bme_sdi),  
    .spi_0_external_SCLK          (bme_sck),  
    .spi_0_external_SS_n          (bme_csn)  
);
```

<図 7. Q2 プロジェクト トップモジュール Card-UNIV6_lcd.v での追加箇所>

図 7 で、赤字で記述した部分が追加した PIO レジスタ部分です。

【QP コンパイル】

QP プロジェクトのトップファイル Card-UNIV6_lcd.v で PD モジュールを記述後は、FPGA のピンアサインを行い、QP コンパイルを実行してください。サンプル回路では FPGA ピンアサインを実施済みです。ピンを追加した場合は、適宜ピンアサインを行ってください。

5. ホスト PC からの制御方法

ホスト PC のボード制御アプリケーション（例えば RefApp7.exe）から、作成した Avalon バスシステムにアクセスすることができます。ただし、PD ツール上で設定したベースアドレスが間違っていると、ボード制御アプリケーションからレジスタアクセスしてもボードは動作しません。ここでは、RefApp7.exe のレジスタアクセスにより、I2C、SPI、LCD 制御を行います。

PD ツールを利用して、バスマスタに GPIF-AVALON ブリッジを使用した場合、PC 上の制御アプリケーションでは、次のように設定して制御できます。

5.1 レジスタアクセス

レジスタ番号と AVALON バス上のアドレスは、下表の通り「レジスタベースアドレス(hex)」+「レジスタ番号 (hex) x 4」で示します。

レジスタ No. (dec)	AVALON バス上のアドレス (hex)
レジスタ 0	レジスタベースアドレス+0x00
レジスタ 1	レジスタベースアドレス+0x04
レジスタ 2	レジスタベースアドレス+0x08
レジスタ 3	レジスタベースアドレス+0x0C
レジスタ 4	レジスタベースアドレス+0x10
.....
レジスタ 8	レジスタベースアドレス+0x20
.....
レジスタ 40	レジスタベースアドレス+0xA0
.....

<表 3. RefApp7 でのレジスタ No.と Avalon バスのベースアドレスとの関連>

※ レジスタベースアドレスは、コンポーネントの登録で設定した値です。

5.2 レジスタ長

8/16/32 ビット・アクセスのみ可能です。GPIF_Master は 32 ビットのデータ幅固定ですので、常に 32 ビットアクセスを行ってください。

ただし、ペリフェラル側で 8/16bit のデータ幅しか無い場合には 32 ビットアクセスではなく、ペリフェラル側のデータ幅に合わせても問題ありません。

5.3 I2C 制御

使用するモジュール名: Avalon I2C(Master) Intel FPGA IP

制御対象センサ: Card-UNIV6 搭載の照度センサ “APDS-9306” スレーブアドレス 0x52

この I2C モジュールのベースアドレスは 0x40 です。I2C モジュール内のメモリマップと RefApp7 から制御するレジスタ No.の対応は以下の通りです。

RefApp7 レジスタ No.	I2C モジュール内 レジスタ No.	内 容
16	0	TRF_CMD : データ通信はこのレジスタを使用
17	1	I2C スレーブからの受信データ
18	2	コントロールレジスタ (未使用)
19	3	インタラプトステータスイネーブルレジスタ (未使用)
20	4	インタラプトステータスレジスタ (未使用)
21	5	ステータスレジスタ (未使用)
22	6	TRF_CMD FIFO レベル確認 (未使用)
23	7	受信データ FIFO レベル確認 (未使用)
24	8	SCL Low 信号設定 (20.8ns x 設定値)
25	9	SCL High 信号設定 (20.8ns x 設定値)
26	A	SDA ホールド時間設定 (20.8ns x 設定値)

<表 4. RefApp7 でのレジスタ No.と I2C モジュール内レジスタ対応表>

I2C で制御する前に、SCL のレートを設定します。この場合、Reg24 と Reg25 に同じ値“0xF0”を WR します。この結果、SCL は約 100KHz の周波数になります。Reg26 に“0xF”を WR すると、SCL 立ち下がりエッジに対して 330ns の SDA データホールド時間を確保できます。

サンプルプロジェクトでは RefApp7 の「TCL スクリプト」タブで利用できるスクリプトファイル “CardUniv6_I2C.tcl” を用意しています。以下に設定の内容を示します。

I2C モジュールの初期設定手順:

1. Reg18 = 0x00 (コアディセーブル、I2C バス速度 100KHz)
2. Reg24 = 0xF0 (SCL Low 時間設定 4992ns)
3. Reg25 = 0xF0 (SCL High 時間設定 4992ns)
4. Reg26 = 0x0F (SDA データホールド時間 312ns)
5. Reg18 = 0x01 (コアイネーブル)

初期設定後、I2C スレーブデバイス（照度センサ）に対してコマンドやデータ RD アクセスができます。

照度センサは以下のフォーマットでアクセスできます。青セルがマスタ出力です。

センサの設定（WR）：

S	SlaveAddr.(7bit)	0	A	RegNo.(8bit)	A	設定データ(8bit)	A.	P
---	------------------	---	---	--------------	---	-------------	----	---

Reg16 = 0x2A4 （スタートビット、スレーブアドレス 0x52、WR）

Reg16 = 0x0□□ （□□：センサのレジスタアドレス）

Reg16 = 0x1△△ （ストップビット、△△：設定データ）

センサデータの読み取り（RD）：

S	SlaveAddr.(7bit)	0	A	RegNo.(8bit)	A	S	SlaveAddr.(7bit)	1	A	データ	N	P
---	------------------	---	---	--------------	---	---	------------------	---	---	-----	---	---

Reg16 = 0x2A4 （スタートビット、スレーブアドレス 0x52、WR）

Reg16 = 0x0□□ （□□：センサのレジスタアドレス）

Reg16 = 0x2A5 （スタートビット、スレーブアドレス 0x52、RD）

Reg16 = 0x100 （ストップビット）

Reg17 = 0x〇〇 （センサから読み出した 8bit データ）

スクリプトファイル“CardUniv6_I2C.tcl”では、照度センサから読み出した 16bit データを 10 進数に変換し、そのデータを 7 セグに書き出すと同時に、読み出したデータが“20000”以上になるまでデータを更新し続けます。

※ センサにスマホのフラッシュライトを近づけると値が 20000 を超えて、読み出し動作を停止します。

I2C アクセス方法の詳細は、インテル社資料[“Embedded Peripherals IP User Guide”](#)を参照してください。

5.4 SPI 制御

使用するモジュール名： SPI (3 Wire Serial) Intel FPGA IP

制御対象センサ： Card-UNIV6 搭載の温湿度・気圧センサ “BME280”

※このセンサは I2C でも制御できます

モジュールの設定は、Master、データ長 16bit MSB ファースト、2MHz クロック、セレクト信号 1 本です。

この SPI モジュールのベースアドレスは 0xA0 です。SPI モジュール内のメモリマップと RefApp7 から制御するレジスタ No.の対応は以下の通りです。

RefApp7 レジスタ No.	SPI モジュール内 レジスタ	内容
40	0 (R only)	Rx データ (8bit)
41	1 (W only)	Tx データ (アドレス 8bit、データ 8bit)
42	2 (R/W)	ステータス (今回未使用)
43	3 (R/W)	制御レジスタ
44	4	未使用
45	5 (R/W)	スレーブ選択 (センサが 1 つなので今回未使用)
46	6 (R/W)	送信ワード数設定 (今回未使用)

<表 5. RefApp7 でのレジスタ No.と SPI モジュール内レジスタ対応表

サンプルプロジェクトでは RefApp7 の「TCL スクリプト」タブで利用できるスクリプトファイル “CardUniv6_BME280.tcl” を用意しています。以下に設定の内容を示します。

BM280 の ID を読み取る

Reg43 = 0x0400 WR (CSn 信号イネーブル)

Reg41 = 0xD000 WR (レジスタアドレス 0xD0)

Reg43 = 0x0000 WR (CSn 信号ディセーブル)

Reg40 RD (ID=0x60 を読み出し)

Reg43[10]ビットにより、CSn を制御し、Reg41 で BME280 レジスタを指定し、アクセスします。

読み出したデータは Reg40[7:0]により得られます。

BME280 には補正用データがレジスタ 0xE1~0xF0、0x88~0xA1 に格納されています。それぞれ読み出して計算できます。

5.5 LCD 制御

使用するモジュール名: PIO (Parallel I/O) Intel FPGA IP

制御対象センサ: Card-UNIV6 搭載の 16x2 キャラクタ LCD (HD44780 コンパチブル)

この PIO モジュールは 16bit 出力設定です。使用する信号は R/S (コマンド/データ種別), E (イネーブル), DB[7:0] (データ) の 10 本です。LCD のデータを読み出すことはできません。LCD に書き込み設定するだけのインタフェースです。

ベースアドレスは 0x80 なので、RefApp7 からは Reg32 を制御します。

RegNo.32(16bit)	9	8	7:0
信号名	R/S	E	DB[7:0]
内容	コマンド/データ種別	イネーブル	データ

<表 6. Reg32 ビット対応表>

サンプルプロジェクトでは RefApp7 の「TCL スクリプト」タブで利用できるスクリプトファイル “CardUniv6_lcd.tcl” を用意しています。このファイルを実行すると、LCD に

Smart-USB Plus
DEMO

と表示します。以下に設定の内容を示します。

LCD の初期設定（コマンド）：

Function set レジスタ 2 回設定

Reg32 = 0x0138

Reg32 = 0x0038 （E 信号の立ち下がりでデータをラッチ）

Reg32 = 0x0000

Reg32 = 0x0138

Reg32 = 0x0038 （E 信号の立ち下がりでデータをラッチ）

Display on/off 設定

Reg32 = 0x010F

Reg32 = 0x000F （E 信号の立ち下がりでデータをラッチ）

Clear display 設定

Reg32 = 0x0101

Reg32 = 0x0001 （E 信号の立ち下がりでデータをラッチ）

Entry mode 設定

Reg32 = 0x0106

Reg32 = 0x0006 （E 信号の立ち下がりでデータをラッチ）

ここまでの設定で LCD の式設定が完了します。

文字を設定するには、以下のレジスタ設定で 1 文字表示ができます。

文字“S”を表示

Reg32 = 0x0353

Reg32 = 0x0253

続けて文字設定すれば、LCD の 1 行目先頭から順番に 16 文字まで表示します。

17 文字以降、表示位置を明示します。改行して 2 行目先頭から文字表示を行うために以下の設定をします。

Reg32 = 0x01C0

Reg32 = 0x00C0

6. 付録

このアプリケーションノートで実現した FPGA 回路の規模は、ロジックエレメント 1,360 個（全体の 9%）、内蔵メモリ 98,376bit（全体の 19%、M9K ブロック 12 個）です。

内蔵メモリは、GPIF_Master モジュールで M9K メモリブロックを 12 個消費します。その他 I2C モジュールで 2 個消費します。