

6-4. ユーザプログラム記述の実例(関数仕様 / プログラム記述 / エラーコード)

御自身の記述したメインプログラムから本ハンドラDLL(+ドライバ)を使用します。

テストには付属のサンプルプログラムを御利用ください。前6-3項に従ってインストールしておきます。本ボードの操作は通常以下の手順となります。具体的なコーディングについてはサンプル・ソースを御覧ください。

- (1)初期化 : 【1】
- (2)サンプリング条件の設定 : 【2】～【6】チャンネル/トリガ/クロック/動作モード
- (3)スタートまたはトリガ待ち : 【7】即スタート/トリガ待ち(内部 or 外部)
- (4)ステータス評価 : 【8】連続サンプリング進捗状況

(5)以下・途中は任意

- (6)サンプリング中止 : 【17】データ転送に割り込み使用時はサンプリング終了後に必須。
- (7)ボードのフラグクリア : 【19】このあと、再度(2) or (3)に戻るときは必要。
- (8)ハンドラ終了 : 【9】後処理

表6-4A. 制御関数一覧

関数名	機能・内容	引数(パラメータ)等
【1】AD_Open_ADSys	ボード、および本ハンドラの初期化	
【2】AD_Set_SampCh	サンプリング実行チャンネル関連設定	チャンネル数、サンプル順、入力範囲
【3】AD_Set_SampMode	サンプリングモード、イベントの設定	ADデータ転送先、方法
【4】AD_Set_Trigger	トリガ関連設定(レンジトリガ以外)	トリガ源、レベル、モード
【4】AD_Set_RangeTrigger	トリガ関連設定(レンジトリガ)	
【5】AD_Set_ExclK	オプション、外部クロック源の設定	クロック源の周波数値
【6】AD_Set_Clock	サンプリング・クロックの設定	クロック源、周期値、単位
【7】AD_Start_Samp	サンプリング開始(トリガ待ち or 即)	サンプリング点数
【8】AD_Get_Status	ステータス取得	サンプリング進捗状況など
【10】AD_Read_DIIData	DLL内バッファからデータ読み出し	ADデータ格納バッファ
【11】AD_Read_DirectFifo	ADボードから直接にデータ読み出し	ADデータ格納バッファ
【12】AD_Get_OneScan	マニュアル(1回)サンプリング	
【9】AD_Close_ADSys	本ハンドラの終了	
	《以上が基本、以下は補助的》	
【13】AD_Out_Aux	汎用デジタル(ラッチ)出力	出力データ
【14】AD_Inp_Aux	汎用デジタル(現在値)入力	入力データ
【15】AD_Set_SampLoop	データバッファをリング状に設定	1廻り後は上書き
【16】AD_Set_Inpmode	信号形式、分解能、データコード指定	
【17】AD_Stop_Samp	サンプリング動作の(強制)中止	
【18】AD_Read_RestData	ADボードの残りデータ読み込み	エラー停止の後
【19】AD_Clear_Flags	ADボードのフラグクリア	ビット指定

重 要

本ハンドラを構成する関数は一定手順、または特定の組み合わせでのみ有効に動作します。 手順については本項先頭に記しましたが、組み合わせの点で注意すべきことは【3】で設定するサンプリングモードと【10】【11】のデータ転送関数です。 以下に連続サンプリング（波形取り込み）の典型的な例を示します。

- <例1> “自動モード”で“スタート操作”を実行すると、当ハンドラ自体が自動的にADボードのFIFOメモリからデータを“当ハンドラDLL内のバッファ”に読み込む。このとき、“ADデータ転送の参照フラグ”を《HALF-FULL》に設定してある場合は“FIFOメモリ容量”の半分単位でブロック転送、また《Not-Empty》に設定の場合は“1スキャン分”単位で転送する。 以上はバックグラウンドで自動実行され、指定データ点数に達すると自動停止する。
- ユーザプログラムでは“サンプリング済みデータ点数”を参照して有効なデータを“当ハンドラDLL内のバッファ”から読み込む。
- <例2> 当ハンドラDLL内のバッファを“エンドレス・リング状”に設定し、例1と同様に操作する。 この場合、“当ハンドラDLL内のバッファ”の末尾と先頭が連結された構造となり、1廻り以後は上書きされる動作となる。
- ユーザプログラム側からは“サンプリング済みデータ点数”の値を当バッファのポインタ（先頭 = 0）として利用する。 連続サンプリングが“ストップ操作”まで無限に実行されるので長時間の監視システムや、低速のプリトリガ動作などに利用できる。
- <例3> “マニュアルモード”で“スタート操作”を実行すると、ADボードが連続サンプリングを開始しても当ハンドラは何もしない。 ユーザプログラムは自力でステータス関数をポーリングして“ボードの生ステータス”から《HALF-FULL》または《Not-Empty》を検出してデータ読み込みタイミングを知り、【11】AD_Read_DirectFifo()で読み込む。
- このとき、“ADデータ転送の参照フラグ”を《HALF-FULL》に設定してある場合は“FIFOメモリ容量”の半分単位でブロック転送、また《Not-Empty》に設定の場合は“1スキャン分”単位で読み込む。
- 当モードでは連続サンプリングが“ストップ操作”まで無限に実行されるので例2と同様なアプリケーションに適用できる。

補足説明

キーワード	説明、関連関数 / 【 】内の数字は関連する関数項目番号
“自動モード”	【3】連続サンプリング動作モード、 および割り込みイベントの設定関数で選択・指定する
“マニュアルモード”	
“ADデータ転送の参照フラグ”	
“スタート操作”	【7】連続サンプリング・スタート関数
“当ハンドラDLL内のバッファ”	ユーザプログラム側へは【10】AD_Read_DIIData()で転送、 容量は【2】指定チャンネル数×【7】指定データ点数 / ch
“FIFOメモリ容量”	【1】初期化のとき自動検出
“1スキャン分”	【2】指定チャンネル数を各1回サンプリング分
“サンプリング済みデータ点数”	【8】ステータスから得る
“エンドレス・リング状”	【15】DLLバッファをリング状・エンドレスに設定関数
“ストップ操作”	【17】連続サンプリング動作の強制停止関数の実行

《追伸》 ユーザプログラム上でADボード上のFIFOメモリ容量をソフト認識したいときは<例3>を“マニュアルモード”“ADデータ転送の参照フラグ”を《HALF-FULL》とし、ユーザプログラム上のバッファ容量を変化させて実行したとき、同容量がFIFOメモリの半分未満なら【11】AD_Read_DirectFifo()実行がエラーとなることから知ることができる

以下に各関数の仕様・詳細を記します。

【1】初期化

書式	int AD_Open_ADSystem (HWND Owner , int address, int intr_no)
引数	Owner : ウィンドウハンドル / 割り込み発生、サンプリング終了、データロス発生等の / / メッセージングに使用。 address : ADボードのベースアドレス / 標準設定 : 0x01d0 (1-3項参照) intr_no : 割り込み番号 / 選択肢 : 3, 5, 7, 9, 10, 11, 12, 15 (3-6項参照)
戻り値	正常終了時 : ボードのID / ADM - 652 AT : 52H、 ADM - 656 AT : 56H エラー時 : エラーコード / 負の値 (エラーコード表)
機能・動作	本ボード上で設定したI/Oアドレスを本ハンドラが認識すると共に、 本ボードのリセット、ハンドラ内部の参照テーブルやデータバッファを初期化する。 また、ADボード上のFIFOメモリ容量をチェックする。

【2】サンプリング実行チャンネル関連の設定

書式	int AD_Set_SampCh (int no_ch, int scan_order[], int range[])
引数	no_ch : サンプリングを実行するチャンネル数。 scan_order[] : 各チャンネルのスキャン順。(未使用: 本ボードでは固定 = 若い番号順) range[] : 各チャンネルの入力レンジ番号 (未使用: 本ボードではスイッチ設定)
戻り値	正常終了時 : = 0 エラー時 : エラーコード / 負の値 (エラーコード表)
機能・動作	サンプリングを実行するチャンネル数、各チャンネルのスキャン順、入力レンジを設定する。 例えばno_ch : 5ならば、スキャン順番0 ~ 4のチャンネル群がサンプリング実行対象となる。 本ボードではスキャン順固定 (0, 1, 2, …… 15)、入力レンジはボード上のスイッチ設定。

【3】サンプリング動作モード、および割り込みイベントの設定

書式	int AD_Set_SampMode (int trs_trig, int buf_area, int intr_sw)
引数	trs_trig : ADデータ転送の参照フラグ / 0 : EMPTY解消、 1 : HALF-FULL buf_area : ADデータ転送先 / 0 : DLL内バッファ、 1 : ユーザプログラム内バッファ (自動モード) (マニュアルモード) intr_sw : 割り込みイベント発生要因指定 / 0 : 割り込み不使用、 1 : 連続サンプリング・クロック 2 : 外部割り込み入力信号 () 3 : 外部割り込み入力信号 () 4 : トリガ発生 5 : 1回サンプリングスキャン終了 6 : Not-Empty 7 : Half-Full
戻り値	正常終了時 : = 0 エラー時 : エラーコード / 負の値 (エラーコード表)
機能・動作	buf_areaでDLL内バッファとした場合は当ハンドラ自体がDLLバッファ内にデータを 読み込む自動モード、ユーザプログラム側からはステータス取得関数でサンプリング済みデータ点数を 認識して専用の関数AD_Read_DLLDataで読み出す。 ユーザプログラム内バッファとした場合はユーザプログラム自体がステータス取得関数でボード上の FIFOメモリ状態を直接監視して、専用関数AD_Read_DirectFifoでユーザプロ グラム側が用意したバッファに読み込むマニュアルモード。 【注1】: いずれのモードでも参照するフラグをtrs_trigで指定する。 【注2】: いずれのモードでもADデータ転送自体には割り込みは使用されない。 また、intr_swは独立したイベント発生要因指定。

【4】トリガ関連の設定A（レンジトリガ以外の場合）

書式	int AD_Set_Trigger(int trg_mode, int trg_source, int trg_pol, int trg_level)
引数	trg_mode : トリガ動作モード / 0 : 即トリガ、1 : ポストトリガ trg_source : トリガ源 / 0 : ソフト、1 : 内部（アナログ）、2 : 外部 trg_pol : トリガ極性 / 0 : 負エッジ、1 : 正エッジ 2 : 負レベル、3 : 正レベル trg_level : トリガレベル（アナログ） / 対応するADデータコード上位8ビットで指定する。
戻り値	正常終了時 : = 0 エラー時 : エラーコード / 負の値（エラーコード表）
機能・動作	トリガ動作モード、源、極性、レベル等の設定。 なお、トリガ動作モードをポストトリガ、トリガ源を外部（デジタル）で極性をレベルに指定したときは帯域サンプリングとなる。（3-1項参照） アナログトリガ源はチャンネル0入力（固定）。

【4】' トリガ関連の設定B（レンジトリガの場合）

書式	int AD_Set_RangeTrigger(int trg_sel, int trg_level_hi, int trg_level_lo)
引数	trg_sel : トリガ動作モード / 0 : アウトレンジ、1 : インレンジ 2 : デュアルスロープ（+）、3 : デュアルスロープ（-） trg_level_hi : トリガレベル上限値 trg_level_lo : トリガレベル下限値
戻り値	正常終了時 : = 0 エラー時 : エラーコード / 負の値（エラーコード表）
機能・動作	レンジトリガの極性を含む動作モード、レベルの設定。（レンジトリガ以外は前記【4】を使用） アナログトリガ源はチャンネル0入力。（固定）

【5】オプション、または外部クロック源周波数値の設定

書式	int AD_Set_Exclk (int exclk_freq)
引数	exclk_freq : オプション、または外部クロック源の周波数値（Hz単位）
戻り値	正常終了時 : = 0 エラー時 : エラーコード / 負の値（エラーコード表）
機能・動作	オプション、または外部クロック源を使用するとき、その周波数値を設定する。 後記【6】ad_set_clockで分周比によるサンプリングクロック指定、またはボードに標準搭載の内部クロック源を使用するときは必要ない。