

【6】サンプリング（波形出力）スタート 《波形データバッファが32K語以内のとき》

<pre>int da_start_samp(WORD num_data, WORD num_block, int upd_mode, WORD *bufptr)</pre>	
num_data	1ブロック当りのデータ数。（サイクルモードでは1周期分：FIFOメモリ容量以内）
num_block	出力ブロック数。（num_data）×（num_block）＝総出力データ数。
upd_mode	更新モード。/ 0：サイクル（停止操作まで無限）、 1：サイクル（ x を出力） 2：非サイクル（ x を出力）
*bufptr	波形データバッファのポインタ。
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5-3参照）
機能・動作	サンプリングをスタート（外部トリガ指定のときはトリガ待ち）させます。 引数は複数チャンネル（複数ボード）使用時も全チャンネル共通です。 データバッファ（パソコン側）は全チャンネル分で、図5-3の構成です。なお、 【3】da_set_sampmodeでbuf_areaを拡張（EMS，XMS）メモリに指定したときは が無視されます。（ により本ハンドラが自動設定）

【6】‘ サンプリング（波形出力）スタート 《波形データバッファが32K語を超えるとき》

<pre>int da_start_samp_h(WORD num_data, WORD num_block, int upd_mode, WORD __huge *bufptr)</pre>	
num_data	1ブロック当りのデータ数。（サイクルモードでは1周期分：FIFOメモリ容量以内）
num_block	出力ブロック数。（num_data）×（num_block）＝総出力データ数。
upd_mode	更新モード。/ 0：サイクル（停止操作まで無限）、 1：サイクル（ x を出力） 2：非サイクル（ x を出力）
*bufptr	波形データバッファのポインタ。
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5-3参照）
機能・動作	サンプリングをスタート（外部トリガ指定のときはトリガ待ち）させます。 引数は複数チャンネル（複数ボード）使用時も全チャンネル共通です。 データバッファ（パソコン側）は全チャンネル分で、図5-3の構成です。なお、 【3】da_set_sampmodeでbuf_areaを拡張（EMS，XMS）メモリに指定したときは が無視されます。（ により本ハンドラが自動設定）

《注1》 サンプリング（波形出力）スタート関数【6】と【6】‘ は総出力データ量の大きさにより使い分けてください。 機能は全く同一です。

《注2》 サイクルモード時の1ブロック当りデータ数（ ）は、FIFO容量（標準4K語）以内で、指定回数（ ）または停止操作まで繰り返し出力します。一方、非サイクルモード時は総出力データ（ x ）を1回だけ出力して動作終了となります。

【7】D AチャンネルA, B即時・同期更新出力

<code>int da_out_ab(int board_no, int upd_va, int upd_vb)</code>	
<code>board_no</code>	更新出力する対象ボードの番号 / 0 : 1 枚目 (マスタ)、 1 ~ 7 : スレーブ
<code>upd_va</code> <code>upd_vb</code>	チャンネルA出力データ。 / 整数 0 ~ 4 0 9 5 (digit) チャンネルB " " / " " (")
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 5 - 3 参照)
機能・動作	チャンネルA, B同時に新データで更新されます。 本関数は【1】初期化の後、単独で実行可能です。 波形出力中に本関数が実行されると、その時点で波形出力が停止し、続いて本関数の結果が得られます。乗算、または減算 (加算) 出力モードのときは、この過程で出力に (瞬間的な) 乱れが生じることがありますから御注意ください。

【8】D AチャンネルBのみ即時更新出力 《チャンネルAが波形出力中の時だけ有効》

<code>int da_out_b(int board_no, int upd_vb)</code>	
<code>board_no</code>	更新出力する対象ボードの番号 / 0 : 1 枚目 (マスタ)、 1 ~ 7 : スレーブ
<code>upd_vb</code>	チャンネルB出力データ。 / 整数 0 ~ 4 0 9 5 (digit)
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 5 - 3 参照)
機能・動作	チャンネルAが波形出力中のとき、チャンネルBが新データで更新されます。 本関数はチャンネルAの波形出力が行われていない時は無効です。当チャンネルBがチャンネルAの振幅制御 (乗算モード) に使用されている場合、またはチャンネルAの波形出力中に当チャンネルBを更新したい場合の用途です。 本関数実行によるチャンネルBの更新タイミングは、直後の (波形出力用) クロック、すなわちチャンネルA更新タイミングです。クロック周期が長い場合や外部クロックを使用するときは注意が必要です。

【9】ボードステータスの取得

<code>int da_get_status(int board_no, WORD *cnt0_data, WORD *cnt1_block, int eos, int *intr_req, int *upd_err, int status[])</code>	
<code>board_no</code>	ステータス取得する対象ボードの番号 / 0 : 1 枚目 (マスタ)、 1 ~ 7 : スレーブ
<code>cnt0_data</code> <code>cnt1_block</code> <code>eos</code> <code>intr_req</code> <code>upd_err</code> <code>status[0]</code> <code>status[1]</code>	(1ブロック中の) 出力データ数カウンタの現在値。 / カウンタ # 0。 出力ブロック数カウンタの現在値。 / カウンタ # 1。 D A 出力終了。 / 1 : 終了、 0 : 出力中または開始前。 割り込み要求発生。 / 1 : 発生、 0 : 未発生 / (3 - 10 項) 更新出力エラー。 / 1 : 発生、 0 : 未発生 / (3 - 18 項) ボードステータス 1。 / 1 バイト生データ (3 - 18 項) ボードステータス 2。 / 未使用
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 5 - 3 参照)
機能・動作	は動作進行状況、または終了の検出に使用します。 なお は一旦セットされると【1】初期化、または【18】フラグクリア関数でリセットするまで保持されます。

【10】サンプリング（波形出力）動作の強制停止

<code>int da_stop_samp(void)</code>	
引 数	な し。
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 5 - 3 参照）
機能・動作	<p>全チャンネル（ボード）のサンプリング動作を強制的に停止します。</p> <p>各チャンネルの出力は最後に更新された値を保持しています。 この後、スタート関数により再スタート（続きを実行）することができます。</p> <p>F I F Oメモリ内の残りデータはそのまま待機状態です。 この後、新たな波形出力を行うときは再度【1】～【5】の手順を踏むか、同一条件で再設定の関数【18】を実行してからスタート【6】or【6】'させます。</p>

【11】本ハンドラの終了

<code>int da_close_dasys(void)</code>	
引 数	な し。
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 5 - 3 参照）
機能・動作	<p>本ハンドラの終了。 確保したメモリ領域の開放等、開始前の状態に戻す。</p> <p>【注】 D A出力の最終状態を維持するためにボードのリセットは行いません。 0 v出力に戻して終了したいときは【7】即時出力関数で0 v相当データを書き込んでから本関数を実行します。</p>

以上【1】～【11】が主要関数です。 以下の【12】～ は補助的な関数です。

【12】拡張メモリにD Aデータを書き込む (h u g e ポインタを使用しない場合)

<pre>int da_write_exmem(int board_no, WORD num_data, WORD num_block, int upd_mode, WORD *bufptr)</pre>	
<pre>board_no num_data num_block upd_mode</pre>	書き込み対象チャンネル（ボード番号：0～7） 1ブロック当りのデータ数 全ブロック数 D/A出力更新モード / 0：サイクル（停止操作まで無限）、 1：サイクル（× を出力）、 1：非サイクル（× を出力）
<pre>bufptr</pre>	バッファのポインタ
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 5 - 3 参照）
機能・動作	<p>標準メモリに入りきれない大容量のデータをD/A出力するときは、当関数を使用して拡張メモリに書き込んで使用します。</p> <p>総データ数を標準メモリ上のバッファに入る程度の複数ブロックに等分し、1ブロックずつバッファに書き込んでから当関数を実行する（ブロック数だけ繰り返す）。 なお、別データを再書き込みするときは本ハンドラを一旦終了、再オープンして行う。</p>

【12】‘拡張メモリにD Aデータを書き込む (h u g e ポインタを使用する場合)’

<pre>int da_write_exmem(int board_no, WORD num_data, WORD num_block, int upd_mode, WORD huge *bufptr)</pre>	
<pre>board_no num_data num_block upd_mode</pre>	書き込み対象チャンネル（ボード番号：0～7） 1ブロック当りのデータ数 全ブロック数 DA出力更新モード / 0：サイクル（停止操作まで無限）、 1：サイクル（× を出力）、 1：非サイクル（× を出力）
<pre>bufptr</pre>	バッファのポインタ
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 5 - 3 参照）
機能・動作	<p>標準メモリに入りきれない大容量のデータをDA出力するときは、当関数を使用して拡張メモリに書き込んで使用します。</p> <p>総データ数を標準メモリ上のバッファに入る程度の複数ブロックに等分し、1ブロックずつバッファに書き込んでから当関数を実行する（ブロック数だけ繰り返す）。 なお、別データを再書き込みするときは本ハンドラを一旦終了、再オープンして行う。</p>

【13】DAデータコードの指定

<pre>int da_set_datacode(int data_code_a, int data_code_b)</pre>	
data_code_a	チャンネルAのDAデータ・コードの選択。 / 0 : バイナリ、 1 : 2の補数
data_code_b	チャンネルBのDAデータ・コードの選択。 / 0 : バイナリ、 1 : 2の補数
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 5 - 3 参照)
機能・動作	本ボードMDA - 7 6 1 A Tに入力コード (DAデータの型) を指定する。 チャンネルA , B ごとに全ボード共通。

【14】クロック / SYNC ・外部出力の切り替え

<code>int da_sel_outsig(int out_sig[])</code>	
<code>out_sig[]</code>	クロック / SYNC ・出力選択。 / 0 : クロック、 1 : SYNC [] 内はボード番号 0 ~ 7、複数ボード使用時の [0] はマスタなので必ず 0 とする。
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 5 - 3 参照)
機能・動作	外部へのクロック信号出力 (コネクタ CN2 の CLK - OUT) を必要なら SYNC 信号 (各 1 ブロック・データの開始タイミング) に切り替えることができる。 但し、複数ボード使用時のマスタはクロックを出力しなくてはならないので、必ずクロック側に設定すること。

【15】 《 BREAK キー押下によるトリガ待ち停止 》の設定 / 解除

<code>int da_onkey_quit(int set_mode)</code>	
<code>set_mode</code>	動作の指定。 / 0 : 解除、 1 : 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 5 - 3 参照)
機能・動作	BREAK (= CTRL + PAUSE) キーの押下により、トリガ待ちを停止させる。サンプリング (波形出力) 中であればサンプリングを強制停止させる。 スタート関数の実行によって外部トリガ待ち状態になったにもかかわらず何かの理由でトリガが入力されない場合、人為的にトリガ待ちループから抜け出る手段となる。

【16】 《 有効キー押下による強制的・即トリガ動作 》の設定 / 解除

<code>int da_onkey_trg(int act_key)</code>	
<code>act_key</code>	有効な動作 key。 / 0 : 解除、 1 : ESC、 2 : SPACE、 3 : ENTER
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 5 - 3 参照)
機能・動作	有効な動作キーの押下により、強制的に即トリガ動作 (波形出力開始) させる。スタート関数の実行によって外部トリガ待ち状態になったにもかかわらず何かの理由でトリガが入力されない場合、人為的にトリガ待ちループから抜け出る手段となる。

【17】 《 (トリガ待ち) タイムアウト 》の設定 / 解除

<code>int da_set_timeout(WORD set_time)</code>	
<code>set_time</code>	タイムアウトまでの時間 (単位 = 秒)。 / 1 ~ 6 5 5 3 5、 0 : 解除、
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 5 - 3 参照)
機能・動作	スタート関数の実行によって外部トリガ待ち状態になったにもかかわらず何かの理由でトリガが入力されない場合、当時間が経過するとトリガ待ちを停止する。

【18】フラグクリア、およびサンプリング条件の再設定

<code>int da_clear_flags(set_mode)</code>	
<code>set_mode</code>	D A出力条件再設定の有無。 / 0 : フラグクリアのみ。 1 : フラグクリア後、D A出力条件を再設定する。
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 5 - 3 参照)
機能・動作	各フラグをクリアし (3 - 1 8 項)、サンプリング条件を前回と同一に再設定する。 D A出力条件の再設定 : 具体的には【2】～【5】の関数を実行する。 フラグクリアにはF I F Oメモリのリセットが含まれており、停止時の残りデータを破棄して (停止時の出力値を起点に) 新データ群を定義・出力することができる。

【19】割り込み発生時に実行するユーザ関数の設定

<code>int da_onint_func(int intr_source, void interrupt_far *func)</code>	
<code>intr_source</code>	割り込み要求の発生要因。 / 0 : クロック (波形出力の各データ更新タイミング) 1 : 外部割り込み入力 () 2 : 外部割り込み入力 () 3 : トリガ発生 4 : 指定ブロック数の更新出力終了 5 : 各ブロックの先頭データ更新出力時
<code>*func</code>	ユーザ作成関数のポインタ
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 5 - 3 参照)
機能・動作	ユーザが任意に作成した割り込み処理関数の指定。 但し、当指定は【3】波形データ転送モードで、割り込みを不使用に設定した場合のみ可能です。 なお、【1】初期化実行の直後は可能な状態です。

【20】本ハンドラのバージョン取得

<code>int da_get_libver(int ver)</code>	
<code>ver</code>	0 : 戻り値は (メジャー・バージョン番号) + (マイナー・バージョン番号) 1 : 戻り値は (メジャー・バージョン番号) 2 : 戻り値は (マイナー・バージョン番号)
戻り値	正常終了時 : 本ハンドラのバージョン番号 エラー時 : エラーコード (負の値 / エラーコード表 5 - 3 参照)
機能・動作	本ハンドラのバージョン情報を得る 例えばバージョンが1.01の場合、本関数を <code>ver = 0</code> として実行すると戻り値は <code>0 x 1 0 1</code> となります。