

GIDADC.(H/CPP) クラスライブラリ

ファイル

GIDADC.H 定義ファイル
GIDADC.CPP プログラム本体
I032.(H,CPP,OBJ) 32ビット対応用の直接I/Oモジュール

コンパイラ

ボーランドのBC++(Ver4~5),Cビルダーでコンパイルできる.

コンパイルSWと注意

__WIN32__ コンパイラがターゲットに応じて設定する
PCAT PC/ATで動作させるときにユーザーが設定する
PC98 PC98で動作させるときにユーザーが設定する

I032(H/CPP/OBJ) 32ビットモードでは,I032.CPPをコンパイルするのにTASM32が必要です.これはコンパイラキットに標準では付属していないので,同時に提供されているコンパイル済みのI032.OBJを統合環境に指定してください.

構築

```
#include "gidadc.h"
GidADC ADC1;
GidADC *ADC2;
class X {
public:
    GidADC ADC3;
    GidADC *ADC4;
    X(void) { ADC2 = new GidADC(); ADC4 = new GidADC(); }
    ~X(void){ delete ADC2; delete ADC4; }
};
```

ADC1 プログラムが立ち上がったときに構築される

ADC2 Xのコンストラクタで構築される

ADC3 Xの構築と同時に構築される

ADC4 Xのコンストラクタで構築される

構築しただけではまだ動作状態にならず ComPort, InitPowerなど必要な初期化をして動作するようになります.

デストラクタはプログラムが完全に終了する場合には自動的に呼び出されます.

変数

```
int Comport;      RS232C COMポート番号 1~4(PCAT) 1(PC98)
int Pola;        アナログ変換モード UNIPOLA(8)=0~4095mV BIPOLA(0)=±2047mV
int Ch;          アナログ入力チャンネル 0~7
int SgIDif;      アナログ入力モード Diff(0)=差動入力 Single(4)=普通の入力

//GetNTCnv() のトリガモード
int TrigaPola;   //1=負から正へ、-1=正から負へ
int TrigaAuto;   //0=none, 1=Auto、2=Normal

int CliMode;    //ブロック変換のときに割り込みを禁止するフラグ
int TxStat;     //Txの制御状態記憶
unsigned char ComMem; //Txの制御状態記憶補助 PC98のみで使われる
```

関数

```
GidADC(void)      //コンストラクタ、内部メモリのイニシャルだけ

//完全初期化、コンストラクタで呼び出されるので普通は呼び出す必要が無い
void Init(void);

//信号線を設定して電源を供給する、ポート番号は1から始まる番号
void InitPower(int ComNum);

void SetCh(int ch); //アナログ入力チャンネル番号(0~7)以後の操作を指定する

//変換モードの指定
void SetUnipola(void){ Pola = UNIPOLA; } //0~4095mV
void SetBipola(void) { Pola = BIPOLA; } //±2047mV
void SetSingle(void) { SgIDif = Singl; } //ノーマル
void SetDiff(void)   { SgIDif = Diff; } //差動

//トリガモードの指定 ブロックモードでデータ変換するばあいに使われる
enum { TNone, TAuto, TNormal };
void SetTrigaNone(void) { TrigaAuto = TNone; } //トリガ無し
void SetTrigaAuto(void) { TrigaAuto = TAuto; } //自動
void SetTrigaNormal(void){ TrigaAuto = TNormal;} //トリガがあるまで待つ
void SetTrigaPlus(void) { TrigaPola = 1; } //正極性
void SetTrigaMinus(void) { TrigaPola = -1; } //負極性

//タイミング指定 クロック、ストロブ、1データごとの待ちループ回数の指定
void SetClkWait(long t) { ClkW = t; if(ClkW <=0) ClkW=1;}
long GetClkWait(void) { return ClkW; }
void SetStbWait(long t) { StbW = t; if(StbW <=0) StbW=1;}

GIDADC(H/CPP) ライブラリ
```

```
long GetStbWait(void) { return StbW; }
void SetBlkWait(long t) { BlkW = t; if(BlkW <=0) BlkW=1;}
long GetBlkWait(void) { return BlkW; }

//ポートコントロール
void SetTx(int i); //Txポートを(1/0)制御する
void SetTxPulse(long t); //t ループ時間(PC 依拠)だけTx信号を1にする

//現在設定されているアナログチャンネルを読み出しながらdataを超えるまでTxを
//1にする。設定値を超えたら0に戻して帰ってくる。最初にデータ取得して試験する
//のでパルスを出さない場合もある。
void GidADC::SetTxPulseWait(int d);

//1サンプルデータ取得
int GetCnv(void);

//規定数のデータ取得
int GetNCnv(int n, int *buf);

規定数のデータ取得、トリガつき .wnはnの整数倍 2 なら2倍、自動モードならこの倍数
だけトリガを待つ.変換中の割り込みは SetCliMode() による
帰り引数 0 なら成功、それ以外ならトリガが失敗したがデータは取得
int GetNTCnv(int v, int n, int wn, int *buf);

PCシステム割り込みを GetNCnv() GetNTCnv() について禁止する設定 他のモードでは
禁止されない。モード設定するとデータ取得間隔はきちりと規定されるが副作用とし
てPCの時間が遅れる、時間遅れは再起動すれば元に戻る。
void SetCliMode(int i = 1) //引数無しならCLIモード
void ReSetCliMode(void) //解除

//時間関数 GetN(T)Cnv の1データあたりの時間を秒単位で調べる
double GetNCnvTime(void); //秒単位
double NCnvTime; //結果を保存しているので後で読み出すこともできる

以下は内部使用
private:
long ClkW; //クロック待ちループの設定 既定値=1
long StbW; //ストロブ待ちループの設定 既定値=1
long BlkW; //GetN(T)Cnv データ間待ちループの設定 既定値=1

void ClkWait(void);
void StbWait(void);
void BlkWait(void);
void SCLK(int Dout); //DATA out Only CLK
int SCLKD(void); //データ読み出しCLK
GIDADC(H/CPP) ライブラリ
```

```
int DOut(int Command); //1bit DOUT  
void SetTxE(void); //Txeポートのコントロール  
void SetComport(int c); //1から始まるCOMポート番号
```

(C)数理設計研究所 1997/9/14