PCI-1202/1602/180x

DLL 使用说明手册

承诺

郑重承诺：凡泓格科技股份有限公司产品从购买即日起一年内无任何材料性缺损。

免责声明

凡使用本系列产品除产品质量所造成的损害，泓格科技股份有限公司不承担任何法律责任。泓格科技股份有限公司有义务提供本系列产品可靠而详尽资料，但保留修订权利，且不承担使用者非法利用资料对第三方所造成侵害构成的法律责任。

版权

商标

手册中所涉及所有公司商标，商标名称及产品名称分别属于该商标或名称的拥有者所有`。

日期：2007/05/01

目　　录

1. 导读

PCI-1202/1602/180x 的开发套件中提供动态链接库(Dynamic Link Library)，以及简明易懂的范例程序。参考这些范例程序，您可以快速的使用 DLL 中的函数完成控制程序，不再需要面对繁琐的硬件缓存器来规划步骤。

本文件为 PCI-1202 系列、PCI-1602 系列及 PCI-180x 系列的 DLL 函数介绍。您可以使用它来了解 DLL 所提供的函数，使用特定功能(连续读取, 多通道输入)要如何规划程序；也可以在编写控制程序时查阅函数的参数与返回值等信息。

DLL 提供的函数分为测试函数集(Test)、驱动函数集(Driver)、模拟输出函数集(D/A)、模拟输入函数集(A/D)、数字输出/入函数集(DI/O)、多通道模拟输入函数集(Magic Scan)、M_Function 函数集、连续读取函数集(Continuous Capture)及批次读取函数集(Batch Capture)。

1.1 节将这些函数集所属函数简单列表分类 。1.2 节为函数返回值错误码定义表。1.3 节表列了 PCI-1202/1602/180x 在量测不同的模拟输入范围时, 调整信号放大倍率的配置码 。1.4 节简单列出线程优先权的设定值与定义，您可以参考设定批次或连续读取资料时执行的优先权。

1.1 函数定义(Function Define)

使用前请注意下列关键词。以方便您的阅读。

关键词	调用函数前需由使用者设定该参数	使用者调用函数后，会回传参数值
[Input]	有	无
[Output]	无	有

下表函数名称与第二章之后的函数使用介绍，是以 PCI-1202 函数库定义的函数名称为主。PCI-1602 或 PCI-180x 系列板卡 DLL 函数名称与 PCI-1202 函数名称相似， 只需更换红色前置字符串即可。

例如：

P1202_ClearScan(void)➔P1602_ClearScan(void) 或 P180x_ClearScan(void)

区分	函数定义
Test	float P1202_FloatSub2(float fA, float fB);

	short P1202_ShortSub2(short nA, short nB);
	WORD P1202_GetDllVersion(void);
	WORD P1202_GetDriverVersion(WORD *wVxdVersion);
Driver	WORD P1202_DriverInit(WORD *wTotalBoards);
	void P1202_DriverClose(void);
	WORD P1202_GetConfigAddressSpace(WORD wBoardNo, WORD *wAddrTimer,WORD *wAddrCtrl, WORD *wAddrDio, WORD *wAddrAdda);
	WORD P1202_ActiveBoard(WORD wBoardNo)
	WORD P1202_WhichBoardActive(void);
Digital I/O	WORD P1202_Di(WORD *wDi);
	WORD P1202_Do(WORD wDo);
D/A	WORD P1202_Da(WORD wDaChannel, WORD wDaVal);
A/D	WORD P1202_SetChannelConfig(WORD wAdChannel, WORD wConfig);
	WORD P1202_AdPolling(float *fAdVal);
	WORD P1202_AdsPolling(float fAdVal[], WORD wNum);
	WORD P1202_AdsPacer(float fAdVal[], WORD wNum, WORD wSampleDiv);

区分	函数定义
Magic	WORD P1202_ClearScan(void);
	WORD P1202_StartScan(WORD wSampleRateDiv, DWORD dwNum, SHORT nPriority);
	WORD P1202_StartScanPostTrg(WORD wSampleRateDiv, DWORD dwNum, SHORT nPriority);
	WORD P1202_StartScanPreTrg(WORD wSampleRateDiv, DWORD dwNum, SHORT nPriority);
	WORD P1202_StartScanMiddleTrg(WORD wSampleRateDiv, DWORD dwNum, DWORD dwN2, SHORT nPriority);
	WORD P1202_StartScanPreTrgVerC(WORD wSampleRateDiv, DWORD dwNum, SHORT nPriority);
	WORD P1202_StartScanMiddleTrgVerC(WORD wSampleRateDiv, DWORD dwNum, DWORD dwN2,SHORT nPriority);
	void P1202_ReadScanStatus(WORD *wStatus, DWORD *dwLowAlarm, DWORD *dwHighAlarm);
	WORD P1202_AddToScan(WORD wAdChannel, WORD wConfig, WORD wAverage, WORD wLowAlarm, WORD wHighAlarm, WORD wAlarmType);

	WORD P1202_SaveScan(WORD wAdChannel, WORD wBuf[]);
	void P1202_WaitMagicScanFinish(WORD *wStatus, DWORD *dwLowAlarm, DWORD *dwHighAlarm);
	WORD P1202_StopMagicScan();
M_Function	WORD P1202_M_FUN_1(WORD wDaNumber, WORD wDaWave, float fDaAmplitude, WORD wAdSampleRateDiv, WORD wAdNumber, WORD wAdConfig, float fAdBuf[], float fLowAlarm, float HighAlarm);
	WORD P1202_M_FUN_2(WORD wDaNumber, WORD wDaWave, WORD wDaBuf[], WORD wAdSampleRateDiv, WORD wAdNumber, WORD wAdConfig, WORD wAdBuf[]);
	WORD P1202_M_FUN_3(WORD wDaNumber, WORD wDaWave, float fDaAmplitude, WORD wAdSampleRateDiv, WORD wAdNumber, WORD wChannelStatus[], WORD wAdConfig[], float fAdBuf[], float fLowAlarm, float fHighAlarm);
	WORD P1202_M_FUN_4(WORD wType, WORD wDaNumber, WORD wDaWave, float fDaAmplitude, WORD wAdSampleRateDiv, WORD wAdNumber, WORD wChannelStatus[], WORD wAdConfig[], float fAdBuf[], float fLowAlarm, float fHighAlarm);

区分	函数定义
Continuous Capture	WORD P1202_Card0_StartScan(WORD wSampleRate, WORD wChannelStatus[], WORD wChannelConfig[],WORD wCount);
	WORD P1202_Card0_ReadStatus(WORD wBuf[], WORD wBuf2[], DWORD *dwP1, DWORD *dwP2, WORD *wStatus);
	void P1202_Card0_Stop(void);
	WORD P1202_Card1_StartScan(WORD wSampleRate, WORD wChannelStatus[],WORD wChannelConfig[],WORD wCount);
	WORD P1202_Card1_ReadStatus(WORD wBuf[], WORD wBuf2[], DWORD *dwP1, DWORD *dwP2,WORD *wStatus);
	void P1202_Card1_Stop(void);
	WORD P1202_FunA_Start(WORD wClock0Div, WORD wChannel0[], WORD wConfig0[], WORD Buffer0, DWORD dwMaxCount0, WORD wClock1Div, WORD wChannel1[], WORD wConfig1[], WORD *Buffer1, DWORD dwMaxCount1, SHORT nPriority);
Batch Capture	WORD P1202_FunA_ReadStatus(void);
	WORD P1202_FunA_Stop(void);
	WORD P1202_FunA_Get(DWORD *P0, DWORD *P1);
	WORD P1202_FunB_Start(WORD wClock0Div, WORD wChannel0[], WORD wConfig0[], WORD *Buffer0, DWORD dwMaxCount0, SHORT nPriority);
	WORD P1202_FunB_ReadStatus(void);
	WORD P1202_FunB_Stop(void);
	WORD P1202_FunB_Get(DWORD *P0);

1.2 错误码定义(Error Code Define)

函数返回值定义

返回值	定义	叙述
0	NoError	正常
1	DriverHandleError	无效的 **VxD/SYS** 处理。**注 1**
2	DriverCallError	**VxD/SYS** 函数调用错误.。**注 1**
3	AdControllerError	嵌入的控制器发生处理错误. 可能是硬件发生损坏请连络代理商作进一步的处理或送修。
4	M_FunExecError	M_Functions 返回错误码
5	ConfigCodeError	wAdConfig 参数设定码错误
7	HighAlarm	fAdBuf[?]>**fHighAlarm**
8	LowAlarm	fAdBuf[?]< **fLowAlarm**
9	AdPollingTimeOut	硬件计时超过时间发生错误。**注 1**
10	AlarmTypeError	仅有 0/1/2/3/4 是有效的验证值
11	FindBoardError	找不到板卡。请确认板卡是否正确插在 PCI 槽上。
12	AdChannelError	无效的 A/D 频道
13	DaChannelError	D/A 频道必须为频道 0 或是频道 1
14	InvalidateDelay	**dwDelayUs** > 8191
15	DelayTimeOut	延迟时间发生超过时间。
16	InvalidateData	无效的资料。
17	FifoOverflow	FIFO 溢出
18	TimeOut	超时。
19	ExceedBoardNumber	无效的板卡顺序号码。(第一张板卡号码为 0)
20	NotFoundBoard	侦测不到板卡. **注 1**
22	FindTwoBoardError	寻找不到第二张板卡。
23	ThreadCreateError	线程创建错误。
24	StopError	停止错误
25	AllocateMemoryError	内存分配失败。也许系统缺少可以使用的内存。

注 1:请更新你的驱动程序，并且重新启动计算机。
请到此处下载最新的驱动程序 http://www.icpdas.com/download/pci/index.htm

1.3 配置码(Configuration Code)

PCI-1202L/1800L/1802L 配置码

Bipolar/Unipolar	模拟输入讯号范围	放大倍率	Settling Time	Configuration Code
Bipolar	+/- 5V	1	3 us	0x00
Bipolar	+/- 2.5V	2	3 us	0x01
Bipolar	+/- 1.25V	4	3 us	0x02
Bipolar	+/- 0.625V	8	3 us	0x03
Bipolar	+/- 10V	0.5	3 us	0x04
Bipolar	+/- 5V	1	3 us	0x05
Bipolar	+/- 2.5V	2	3 us	0x06
Bipolar	+/- 1.25V	4	3 us	0x07
Unipolar	0V ~ 10V	1	3 us	0x08
Unipolar	0V ~ 5V	2	3 us	0x09
Unipolar	0V ~ 2.5V	4	3 us	0x0A
Unipolar	0V ~ 1.25V	8	3 us	0x0B

PCI-1202H/1800H/1802H 配置码

Bipolar/Unipolar	模拟输入讯号范围	放大倍率	Settling Time	Configuration Code
Bipolar	+/- 5V	1	23 us	0x10
Bipolar	+/- 0.5V	10	28 us	0x11
Bipolar	+/- 0.05V	100	140 us	0x12
Bipolar	+/- 0.005V	1000	1300 us	0x13
Bipolar	+/- 10V	0.5	23 us	0x14
Bipolar	+/- 1V	5	28 us	0x15
Bipolar	+/- 0.1V	50	140 us	0x16
Bipolar	+/- 0.01V	500	1300 us	0x17
Unipolar	0V ~ 10V	1	23 us	0x18
Unipolar	0V ~ 1V	10	28 us	0x19
Unipolar	0V ~ 0.1V	100	140 us	0x1A
Unipolar	0V ~ 0.01V	1000	1300 us	0x1B

PCI-1602 配置码

Bipolar/Unipolar	模拟输入讯号范围	放大倍率	Settling Time	Configuration Code
Bipolar	+/- 10V	1	3 us	0
Bipolar	+/- 5V	2	3 us	1
Bipolar	+/- 2.5V	4	3 us	2
Bipolar	+/- 1.25V	8	3 us	3

1.4 优先权(Priority)

线程的优先权设定

设定值	定义
-2	THREAD_PRIORITY_LOWEST
-1	THREAD_PRIORITY_BELOW_NORMAL
0(Defual)	**THREAD_PRIORITY_NORMAL**
1	THREAD_PRIORITY_ABOVE_NORMAL
2	THREAD_PRIORITY_HIGHEST
15	THREAD_PRIORITY_TIME_CRITICAL
Others	THREAD_PRIORITY_NORMAL

2. 函数介绍

PCI-1202 函数库提供的子程序函数主要可区分为以下 9 个子集:

- 测试 (Test) 函数集
- 驱动 (Driver)函数集
- 数字量 I/O 函数集
- D/A 函数集
- A/D 函数
- A/D 魔术扫描函数集
- M_Function 函数集
- 连续采集函数集
- 批次连续采集函数集

章节 1.1 已详细列出每个子集下包含的函数清单。以下将依序说明这些函数的参数, 功能与可参考的范例程序。

2.1 测试函数集

2.1.0 P1202_FloatSub2

计算 C=A-B(浮点数格式), **float=4个字节的浮点数.** 这个函数目的在测试动态连结函数库(P1202.DLL)是否可以连结, 正确的返回运算的结果.

- **语法:**

  float P1202_FloatSub2(float fA, float fB);

- **参数：**

  fA : [Input] **4** 个字节的浮点数

  fB : [Input] **4** 个字节的浮点数

- **返回值:**

  返回 fA-fB 的值。

- **范例程序：**

  DEMO1.C

2.1.1 P1202_ShortSub2

计算 C=A-B (短整数格式), **SHORT=16 个位有号数.** 这个函数目的在测试动态连结函数库(P1202.DLL)是否可以连结, 正确的返回运算的结果.

- **语法：**

  short P1202_ShortSub2(Short nA, Short nB);

- **参数：**

  nA : [Input] 16 位整数

  nB : [Input] 16 位整数

- **返回值:**

  返回 nA-nb 的值。

- **范例程序：**

  DEMO1.C

2.1.2 P1202_GetDllVersion

读取 P1202.DLL 的 DLL 版本号码.

- **语法：**

  WORD P1202_GetDllVersion(void);
- **参数：**

  无参数
- **返回值：**

  直接返回 DLL 版本，如果返回值等于 0x200， DLL 版本为 2.0
- **范例程序：**

  DEMO1.C

2.1.3 P1202_GetDriverVersion

读取 NAPPCI.VxD 或是 NAPPCI.SYS 的版本。

- **语法：**

  WORD P1202_GetDriverVersion(WORD *wDriverVersion);

- **参数：**

  *wDriverVersion : [output] 传址参数。返回的驱动程序版本号。

  ※wDriverVersion=0x200 → 驱动程序的版本号为 2.0
- **返回值：**

  0 : 无错误

  其它 : 请参考 1.2 节 : 错误码定义。
- **范例程序：**

  DEMO1.C

2.2 驱动函数集

2.2.0 P1202_DriverInit

这个函数会向系统要求分配资源，侦测板卡是否已安装成功，并返回安装板卡的数量。您必须在程序起始处，使用其它函数之前调用这个函数。

- **语法：**

  WORD P1202_DriverInit(WORD *wTotalBoard);

- **参数：**

  *wTotalBoard :[Output] 传址参数，用来储存及返回所寻找到的板卡数量。

  ※wTotalBoard=1 → 系统安装有一张 PCI-1202 板卡
  wTotalBoard=n → 系统安装有 N 张 PCI-1202 板卡

- **返回值：**

  0 : 无错误
  其它 : 请参考 1.2 节 : 错误码定义。

- **范例程序：**

  所有的范例程序.

2.2.1 P1202_DriverClose

释放板卡占用的资源。
这个函数必需在程序结束前调用，将占用的资源释放归还给系统。

- **语法：**
  void P1202_DriverClose(void);

- **参数：**
  无参数

- **返回值:**
  无返回值

- **范例程序：**
  所有范例程序.

2.2.2 P1202_GetConfigAddressSpace

取得第 N 张卡的 I/O 地址。这个函数没有调用不会影响到板卡其它程序的使用。

- **语法：**

  WORD P1202_GetConfigAddressSpace(WORD wBoardNo, WORD *wAddrTimer,WORD *wAddrCtrl, WORD *wAddrDio, WORD *wAddrAdda);

- **参数：**

  wBoardNo: [Input] 用户指定的板卡编号,由 0 算起.
  *wAddrTimer: [Output] 取得 8254 定时器的 I/O 映射地址.
  *wAddrCtrl: [Output] 取得控制器的 I/O 映射地址.
  *wAddrDio: [Output] 取得数字量 I/O 埠的 I/O 映射地址.
  *wAddrAdda: [Output] 取得 A/D 及 D/A 的 I/O 映射地址.
  ※请参考 “PCI-1202/1602/1800/1802 Hardware manual” 第三章 会有详细的说明.

- **返回值：**

  0 : 无错误
  其它 : 请参考 1.2 节 : 错误码定义。

- **范例程序：**

  DEMO1.C

2.2.3 P1202_WhichBoardActive

返回正在启用(Active)状态中的板卡号码。P1202.dll

- **语法:**

  WORD P1202_WhichBoardActive(void);

- **参数:**

  无参数

- **返回值:**

  将正在启用状态中的板卡号码回传出来。如果您使用第二张板卡。将会回传 1 (第一张板卡会回传 0 而不是 1，板卡排序是从零开始)

- **范例程序:**

  DEMO1.C

2.2.4 P1202_ActiveBoard

这个函数会激活(Active)您选用的板卡，就像打开电视的电源开关一样。 当系统安装两张(含)以上的板卡时, 同时间只能设定一张板卡为激活状态。要对非激活状态的板卡进行数字量 I/O、 A/D、 D/A、 M_Function、 魔术扫描及连续采集等程序之前必须先调用这个函数激活 wBoardNo 对应的板卡。

- **语法:**

  WORD P1202_ActiveBoard(WORD wBoardNo);

- **参数:**

  wBoardNo: 要激活的板卡编号 **(从 0 开始，第一张版卡号码为 0).**

- **返回值:**

  0 : 无错误

  其它 : 请参考 1.2 节 : 错误码定义。

- **范例程序 :**

  所有的范例程序.

- **参考资料 :**

  不需要调用 P1202_ActiveBoard()即可运作的函数如下:

P1202_FloatSub2
P1202_ShortSub2
P1202_GetDriverVersion
P1202_DriveInit
P1202_DriveClose
P1202_GetConfigAddressSpace
P1202_Card0_StartScan
P1202_Card0_ReadData
P1202_Card0_Stop
P1202_Card1_StartScan
P1202_Card1_ReadData
P1202_Card1_Stop

2.3 数字量 I/O 函数集

2.3.0 P1202_Di

这个函数会从数字量输入端口读取 16 位的数字量资料回来。它只能作用在激活状态中的板卡，请使用 P1202_ActiveBoard (2.2.4 节) 激活您要使用的板卡。

- **语法：**

  WORD P1202_Di(WORD *wDi);

- **参数：**

  *wDi : [Output] 传址参数。用来储存并返回一个 16 位的 DI(输入)值

- **返回值：**

  0 : 无错误

  其它 : 请参考 1.2 节 : 错误码定义。

- **范例程序：**

  DEMO1.C

2.3.1 P1202_Do

这个函数用来传送一个 16 位的值到数字量输出端口。它只能作用在激活状态中的板卡，请使用 P1202_ActiveBoard(2.2.4 节) 激活您要使用的板卡。

- **语法：**

  WORD P1202_Do(WORD wDo);

- **参数：**

  wDo : [Input]一个 16 位的输出值。

- **返回值：**

  0 : 无错误

  其它 : 请参考 1.2 节 : 错误码定义。

- **范例程序：**

  DEMO1.C

CN1 与 CN2 对接时：
P1202_Do(0x10) ;//输出 0x10
P1202_Di(*wDi) ;//*wDi=0x10 输入 *wDi 值会依接线或外部信号之不同而有所改变。

2.4 DA 函数集

2.4.0 P1202_Da

这个函数会传送一个 12 位的值至模拟输出端口。输出电压的范围由板卡的 JP1 跳线决定是 10V 至-10V 之间或 5V ~ -5V 之间。它只能作用在激活状态中的板卡，请使用 P1202_ActiveBoard(2.2.4 节)函数激活您要使用的板卡。

- **语法：**

  WORD P1202_Da(WORD wChannel, WORD wDaVal);

- **参数：**

  wChannel : [Input] 模拟输出通道.

  wChannel=0 (设定输出到模拟量输出通道 0)

  wChannel=1 (设定输出到模拟量输出通道 1)

  wDaVal : [Input]一个 12 位值将会传送到 DA 端口。此值就是控制模拟输出的电压大小。最小值为 0 最大值为 4095. 模拟输出的范围有二种选择分别为 +/- 5V 或 +/- 10V 可以通过设定版卡上的 JP1 跳线来调整。但是函数无法侦测板卡上 JP1 的设定。所以当 wDaVal 等于 4095 时可能是+10V 也可能是+5V, 是由 JP1 跳线的状态决定。

- **返回值：**

  0 : 无错误

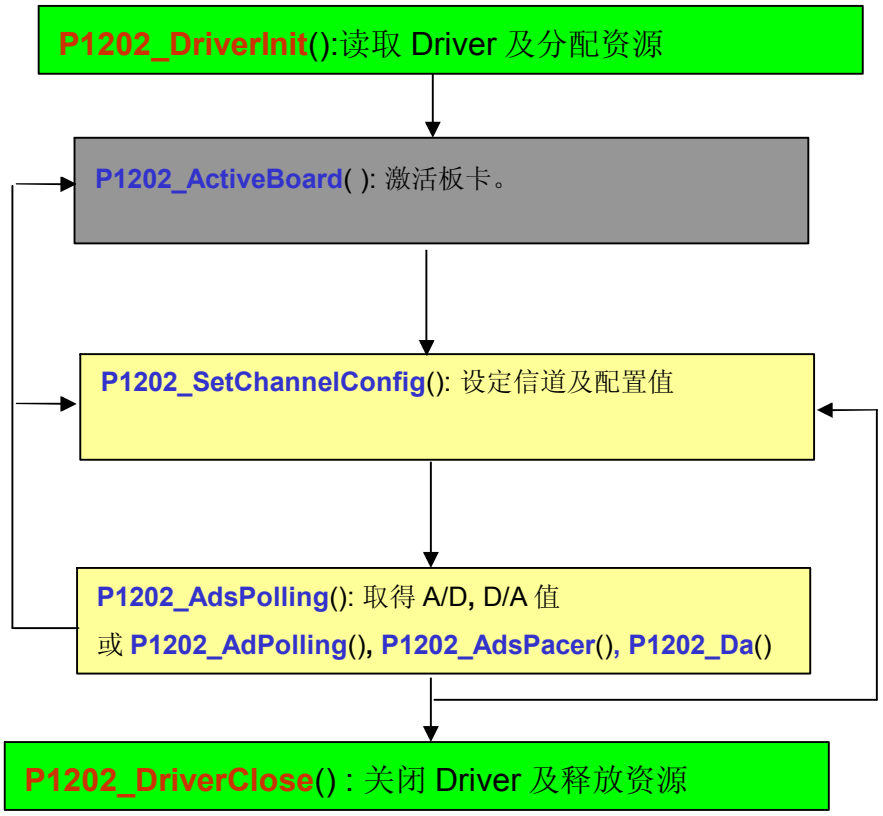
  其它 : 请参考 1.2 节 : 错误码定义。

- **范例程序：**DEMO1.C

2.5 AD 函数集

2.5.0 流程图

AD/DA 轮询函数使用程序



2.5.1 P1202_SetChannelConfig

用来设定各通道的配置值(请参考 1.3 节)。调用 **P1202_AdPolling**, **P1202_AdsPolling** 及 **P1202_AdsPacer** 之前, 须先使用此函数设定要从那一个通道读取数据与合适的配置值(章节.1.3). 它只能使用在激活状态中的板卡，请使用 P1202_ActiveBoard(2.2.4 节) 激活您要使用的板卡。

- **语法：**
  WORD P1202_SetChannelConfig(WORD wChannel, WORD wConfig);
- **参数：**
  wChannel : [Input]选择读取数据的通道。
  wConfig : [Input]设定资料读取范围配置码。请参考 1.3 节。
- **返回值：**
  0 : 无错误
  其它 : 请参考 1.2 节 : 错误码定义。
- **范例程序：**
  DEMO1.C

2.5.2 P1202_AdPolling

这个函数以软件轮询的方式作一次 AD 转换，并返回此转换值。调用这个函数之前需先使用 **P1202_SetChannelConfig**()设定(或改变)通道号码与配置值。P1202_AdPolling 会根据您的设定令指定的通道完成一次 AD 转换，并回传此转换值。它只能使用在激活状态中的板卡，请使用 P1202_ActiveBoard (2.2.4 节) 激活您要使用的板卡。

- **语法：**
  WORD P1202_AdPolling(float *fAdVal);
- **参数：**
  *fAdVal :[Output] 传址参数。用来储存一个 A/D 资料。
- **返回值：**
  0 : 无错误
  其它 : 请参考 1.2 节 : 错误码定义。
- **范例程序：**
  DEMO1.C

2.5.3 P1202_AdsPolling

如果想要取得一个以上的值，使用上一个函数P1202_AdPolling，需要重复多次的调用，将会降低连续读取的效率。所以如果想要连续性的取得AD值的话可以使用这个函数来操作。它会一次将所有转换的的AD值通通存入到buffer里面，这样用起来就方便很多了。**P1202_AdsPolling** 函数利用软件轮询的方式对ADC trigger（触发），使 ADC 开始作转换。调用这个函数之前需先使用**P1202_SetChannelConfig()**设定(或改变)通道及配置值。P1202_AdsPolling 将会根据您的设定来操作。它只能使用在激活状态中的板卡，请使用P1202_ActiveBoard(2.2.4节) 激活您要使用的板卡。

- **语法：**

WORD P1202_AdsPolling(float fAdVal[], WORD wNum);

- **参数：**

fAdVal[]:[Output]是一个浮点数数组。用来储存 ADC 转换完成的数据。每一笔数据与 **P1202_SetChannelConfig()**的设定会有关联性.

wNum:[Input] 由用户指定要取得的数据笔数。

- **返回值：**

0 : 无错误

其它 : 请参考 1.2 节 : 错误码定义。

- **范例程序：**

DEMO1.C

2.5.4 P1202_AdsPacer

这个函数与P1202_AdsPolling()相同，用来取得同一通道上的多笔连续数据，调用的程序也与P1202_AdsPolling()相似。唯一不同的是这个函数透过硬件8254定时器在固定的时间产生触发信号通知AD转换器将模拟信号转换为数字值。

- **语法：**

  WORD P1202_AdsPacer(float fAdVal[], WORD wNum, WORD wSampleDiv);

- **参数：**

  fAdVal[]:[Output] 是一个浮点数数组。用来储存 ADC 转换完成后得到的数据。每一笔数据与 **P1202_SetChannelConfig()**的设定会有关联性

  wNum : [Input] 一次转换几笔 AD 值。

  wSampleDiv: [Input] **AD 取样频率 = 8M/wSampleDiv.**

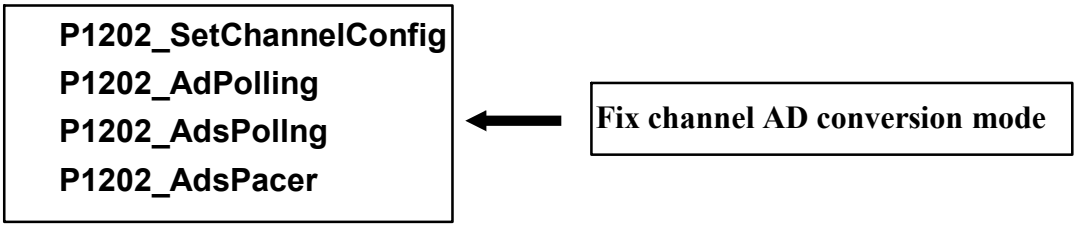
  例如:
  wSampleDiv =80 → 取样频率=8M/80=100K

- **返回值：**

  0 : 无错误

  其它 : 请参考 1.2 节 : 错误码定义。

- **范例程序：**DEMO1.C



2.6 魔术扫描函数集

2.6.0 流程图

•使用魔术扫描的程序:

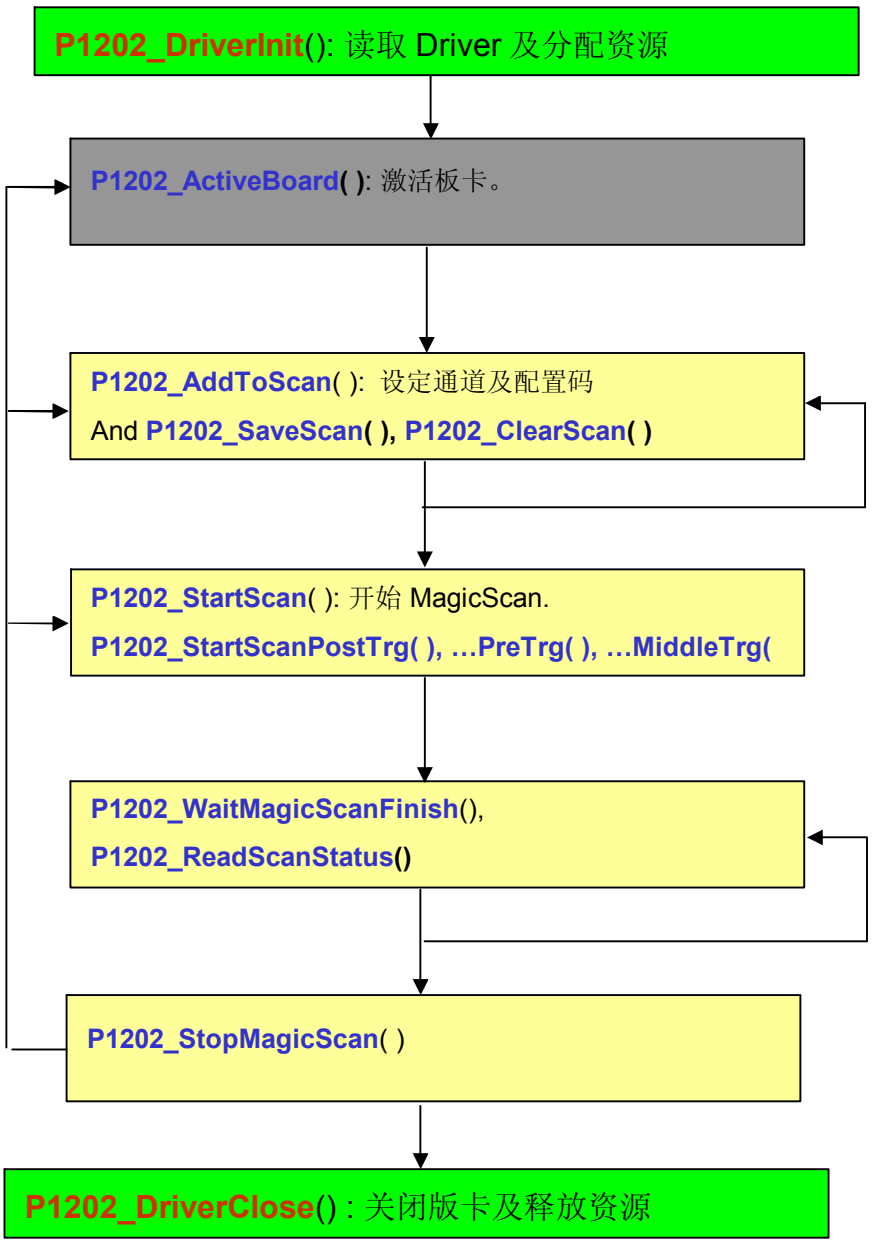
步骤 1. 设定使用魔术扫描的通道扫描顺序与其配置码

步骤 2. 开始魔术扫描

步骤 3. 取得魔术扫描的状态

步骤 4. 取得资料

下图为魔术扫描的函数调用流程图(以 PCI-1202 为例):



2.6.1 P1202_ClearScan

这个函数程序用来重新设定**魔术扫描控制器※**到初始状态。它只能使用在激活状态中的板卡，请使用 P1202_ActiveBoard(2.2.4 节) 激活您要使用的板卡。

※：请参考第三章的注 1。

- **语法：**

  WORD P1202_ClearScan();

- **参数：**

  无参数

- **函数返回值:**

  0 : 无错误

  其它 : 请参考 1.2 节 : 错误码定义。

- **范例程序：**

  DEMO11.C

2.6.2 P1202_StartScan

这个程序用来激活魔术扫描 的运作。您可以使用 **P1202_WaitMagicScanFinish()** 或是 **P1202_ReadScanStatus()** 检查魔术扫描的运作状态。它只能使用在激活状态中的板卡，请使用 P1202_ActiveBoard(2.2.4 节) 激活您要使用的板卡。

- **语法：**

WORD P1202_StartScan(WORD wSampleRateDiv, WORD wNumCycles, , SHORT nPriority);

- **参数：**

wSampleRateDiv :[Input] **A/D 取样频率 = 8M/wSampleRateDiv.**

wSampleRateDiv=80 → 取样频率=8M/80=100K

wNumCycles :[Input] 每一次扫描命令要完成的扫描周数。换句话说，也就是每一次扫描命令完成时每一个通道要取得的资料数

nPriority :[Input] 设定线程的优先权(Priority)。请参考 1.4 节。

- **返回值：**

0 : 无错误

其它 : 请参考 1.2 节 : 错误码定义。

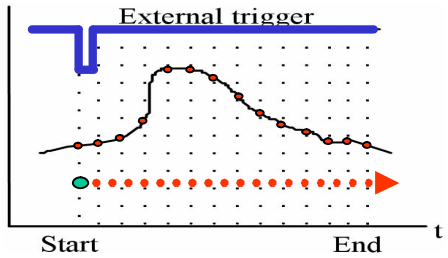
- **范例程序：**

DEMO11.C

2.6.3 P1202_StartScanPostTrg

这个函数使用在当外部触发信号发生时，**魔术扫描控制器**会命令 Timer1 定时触发 AD 转换，简而言之就是先接收到外部信号才会开始抓资料。使用者可以使用 **P1202_ReadScanStatus()** 这个函数检查运作状态。使用 **P1202_StopMagicScan(…)** 函数停止 魔术扫描 运作(强制结束)

※：请参考第三章的注 1。



- **语法：**

WORD P1202_StartScanPostTrg(WORD wSampleRateDiv, WORD wNumCycles, , SHORT nPriority);

- **参数：**

wSampleRateDiv :[Input] **A/D 取样频率 = 8M/wSampleRateDiv.**

wSampleRateDiv=80 → 取样频率=8M/80=100K

wNumCycles :[Input] 每一次扫描命令要完成的扫描周数。换句话说，也就是每一次扫描命令完成时每一个通道要取得的资料数。

nPriority :[Input] 设定线程的优先权(Priority)。请参考 1.4 节。

- **返回值：**

0 : 无错误

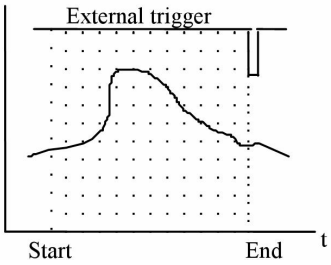
其它 : 请参考 1.2 节 : 错误码定义。

- **范例程序：**

DEMO23.C

2.6.4 P1202_StartScanPreTrg

一开始先以指定的频率将数据取回并存放在 FIFO。此函数会先设定好 Timer 并开始触发读取 AD 转换，直到收到外部触发信号才会结束魔术扫描的动作，然后返回触发之前的 wNumCycles 资料。简而言之就是先抓取资料再接收外部触发信号。使用者可以使用 **P1202_ReadScanStatus() 函数**检查运作状态；使用 **P1202_StopMagicScan(…)** 函数停止 魔术扫描 运作(强制结束)。



- **语法：**

WORD P1202_StartScanPreTrg(WORD wSampleRateDiv, WORD wNumCycles, , SHORT nPriority);

- **参数：**

wSampleRateDiv :[Input] **AD 取样频率 = 8M/wSampleRateDiv.**

wSampleRateDiv=80 → 取样频率=8M/80=100K

wNumCycles :[Input] 每一次扫描命令要完成的扫描周数。换句话说，也就是每一次扫描命令完成时每一个通道要取得的资料数。

nPriority :[Input] 设定线程的优先权(Priority)。请参考 1.4 节。

- **返回值：**

0 : 无错误

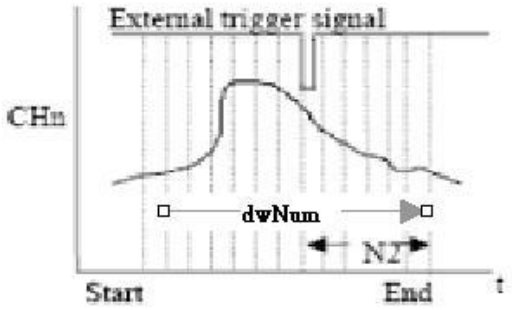
其它 : 请参考 1.2 节 : 错误码定义。

- **范例程序：**

DEMO24.C

2.6.5 P1202_StartScanMiddleTrg

这个函数大致上与上一个函数用法相同，唯一不同之处就是在外部信号触发时会先读取触发后的 N2 笔资料再停止魔术扫描的动作。使用者可以使用 **P1202_ReadScanStatus()** 这个函数去检查运作状态。使用 **P1202_StopMagicScan(…)** 函数来停止 魔术扫描 运作(强制结束)。



- **语法：**

WORD P1202_StartScanMiddleTrg(WORD wSampleRateDiv, dWORD dwNum, WORD wN2 , SHORT nPriority);

- **参数：**

wSampleRateDiv :[Input] **AD 取样频率 = 8M/wSampleRateDiv.**

wSampleRateDiv=80 → 取样频率 =8M/80=100K

dwNum :[Input] 设定要取得外部触发信号在触发之前的资料笔数(per channel)。

WN2 :[Input] 设定要取得外部触发信号在触发之后的资料笔数(per channel)。

nPriority :[Input] 设定线程的优先权(Priority)。请参考 1.4 节。

- **返回值：**

0 : 无错误

其它 : 请参考 1.2 节 : 错误码定义。

- **范例程序：**

DEMO25.C

2.6.6 P1202_ReadScanStatus

这个函数目的在取得 MagicScan、Pre-tirgger、Post-trigger、Mid-trigger 的运作状态。它只能用来检查状态但并不会阻断程序的运作。所以建议您必须随时检查状态，当状态值 0X80 时为正常，若为 0X01 时则代表失败。

它只能使用在激活状态中的板卡，请使用 P1202_ActiveBoard(2.2.4 节) 激活您要使用的板卡。

- 语法：

void P1202_ReadScanStatus(WORD *wStatus, DWORD *dwLowAlarm, DWORD *dwHighAlarm);

- 参数：

*wStatus :[Output] 传址参数。用来储存魔术扫描 的状态。

表 2.8.6.1 wStatus 值状态表

wStatus Value(HEX)	状态	叙述
0x00	initial	MagicScan 初始状态 (发呆中)
0x01	start	MagicScan 工作中
0x02	time out1	MagicScan stage 1 controller 超时
0x04	time out2	MagicScan stage 2 controller 超时
0x08	overflow	MagicScan FIFO overflow※注
0x80	OK	MagicScan 完成

※注：请参考第三章的注 2

*dwLowAlarm :[Output] 传址参数。用来设定 魔术扫描每个通道是否使用 Low alarm 状态.

*dwHighAlarm :[Output] 传址参数。用来设定 魔术扫描每个通道是否使用 High alarm 状态.

dwLowAlarm(bit 0~15)

Bit	F	E	D	C	B	A	9	8	7	6	5	4	3	2	1	0
channel	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

dwLowAlarm(bit 16~31)

Bit	1F	1E	1D	1C	1B	1A	19	18	17	16	15	14	13	12	11	10
channel	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16

dwLowAlarm 是 32 个 bit 的参数直接对应到 32 个通道。

Ch ? = 0 → 无 low alarm

Ch ? = 1 → 有 low alarm

例如:

dwLowAlarm=0 → 所有通道正常，皆无 low alarm

dwLowAlarm=1 → 通道 0 是 low alarm, 其他都正常。

dwLowAlarm=3 → 通道 0、1 是 low alarm, 其他都正常。

dwHighAlarm(bit 0~15)

Bit	F	E	D	C	B	A	9	8	7	6	5	4	3	2	1	0
channel	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

dwHighAlarm(bit 16~31)

Bit	1F	1E	1D	1C	1B	1A	19	18	17	16	15	14	13	12	11	10
channel	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16

dwHighAlarm 是 32 个 bit 的参数直接对应到 32 个通道。

Ch ? = 0 → 无 high alarm

Ch ? = 1 → 有 high alarm

Example:

dwHighAlarm=0 →所有通道正常，皆无 high alarm

dwHighAlarm=4 → 通道 2 是 high alarm, 其他都正常。

dwHighAlarm=7 → 通道 0、1、2 是 high alarm,其他都正常。

- ## 返回值:

无返回值

- ## 范例程序：

DEMO11.C

2.6.7 P1202_AddToScan

这个函数用来新增一个通道到**魔术扫描**循环扫描队列。**魔术扫描**循环扫描队列最多可以增加到 48 个通道。调用这个函数增加通道的顺序极为扫描的顺序，例如：

```
P1202_AddToScan(1, 0, 1, 0, 0, 0);
P1202_AddToScan(1, 0, 1, 0, 0, 0);
P1202_AddToScan(0, 0, 1, 0, 0, 0);
```

则循环扫描顺序为 ch1 -> ch1 -> ch0 -> ch1 -> ch1 -> ch0 -> ch1

它只能使用在激活状态中的板卡，请使用 P1202_ActiveBoard(2.2.4 节) 激活您要使用的板卡。

- **语法：**

word P1202_AddToScan(WORD wAdChannel, WORD wConfig, WORD wAverage, WORD wLowAlarm, WORD wHighAlarm, WORD wAlarmType);

- **参数：**

wAdChannel :[Input] A/D 通道号。
wConfig :[Input] 配置码。请参考 1.3 节。
wAverage :[Input] 数字平均筛选值。在一个平均数量里作出平均值。例如：读取 10 个 AD 值作平均计算出来的平均值。最小值为 1。

> (**数字平均筛选值**)如果将 wAverage 设为 5。你将会得到一个值，这一个值是由五个读取值所平均计算出来的。所以取样频率也会只剩下五分之一。

wLowAlarm :[Input] 12 bits low alarm 资料
wHighAlarm :[Input] 12 bits high alarm 资料
wAlarmType :[Input] Alarm 型态.

表 2.8.7.1 Alarm 类型定义表

AlarmType Value	Alarm Type	描述
0	No alarm	关闭
1	High alarm	范围以上
2	Low alarm	范围以下
3	In alarm	介于之间
4	Out alarm	在范围之外

范例:

P1202_AddToScan(0, 0, 1, 0, 0, 0)

wAdChannel =0→ 开启通道 0 加入扫描

wConfig = 0→设定 A/D 范围为 5V~ -5V

wAverage=1→不作平均

wLowAlarm=0 →无 LowAlarm

wHighAlarm=0 →无 HighAlarm

wAlarmType=0→关闭 Alarm

- 返回值：

  0：无错误

  其它：请参考 1.2 节：错误码定义。

- 范例程序：DEMO11.C

2.6.8 P1202_SaveScan

这个函数用来指定循环扫描取得的数据储存的位置。须先为每一个加入循环扫描的通道宣告适当大小的内存空间，再用这个函数指定循环扫描队列中每个通道的资料要放在哪一个内存中。

- **语法：**

  void P1202_SaveScan(WORD wAdChannel, WORD wBuf[]);

- **参数：**

  wAdChannel :[Input] 循环扫描顺序号码，从 0 开始。(注意：非通道号码)。
  例如：第一个被扫描的通道，可能是任一个实体通道，但在这个函数中 wAdChannel 都是 0，因为这个参数是扫描的顺序号码。
  wBuf :[Output] 储存 wADChannel 取回资料的内存。

```
范例:

    WORD   wV0[100000];         // AD ch:0 buffer
    WORD   wV2[100000];         // AD ch:4 buffer
     :
     wRetVal=P1202_ClearScan();
   //****    For PCI-1202L
    wRetVal += P1202_AddToScan(0,0,1,0,0,0);       // add CH:0 to scan
    wRetVal += P1202_SaveScan(0,wV0);
     /*CH0 为第一个加入扫描的通道, 扫描顺序号码为 0, 将扫描顺序号码为 0 这个通
     道的资料存入 wV0[ ] buffer 里*/

     wRetVal += P1202_AddToScan(4,0,1,0,0,0);      // 将通道 4  加入循环扫描队列
    wRetVal += P1202_SaveScan(1,wV2);
     //注意：1 => 第二个被扫描的信道，而非实体信道号码 4
     /* CH4 是第二个加入扫描顺序的通道, 扫描顺序号码为 1, 将扫描顺序号码为 1 的
      这个信道的数据存入 wV2[ ] buffer 里*/

    wSampleRateDiv=80;            //      取样频率=8M/80 = 100k
    P1202_StartScan(wSampleRateDiv,DATALENGTH,nPriority);    // 开始扫描
```

- **返回值：**

  0：无错误
  其它：请参考 1.2 节：错误码定义。

- **范例程序：**.DEMO11.C

2.6.9 P1202_WaitMagicScanFinish

这个程序用作搁置模式 (blocking mode) 处理及等待持续到魔术扫描运作停止为止。

它只能使用在激活状态中的板卡，请使用 P1202_ActiveBoard(2.2.4 节) 激活您要使用的板卡。

- **语法：**

void P1202_WaitMagicScanFinish(WORD *wStatus, DWORD *dwLowAlarm, DWORD *dwHighAlarm);

- **参数：**

wStatus :[Output] 传址函数。用来检查魔术扫描 的状态。

Table 2.8.6.1 wStatus 值状态表

wStatus Value(HEX)	状态	叙述
0x00	initial	MagicScan 初始状态 (发呆中)
0x01	start	MagicScan 开始运作
0x02	time out1	MagicScan stage 1 controller 超时
0x04	time out2	MagicScan stage 2 controller 超时
0x08	overflow	MagicScan FIFO overflow※注
0x80	OK	MagicScan 完成

※注：请参考第三章的注 2

*dwLowAlarm :[Output] 传址参数。用来检查 魔术扫描每个通道是否处于 Low alarm 状态.

*dwHighAlarm :[Output] 传址参数。用来检查 魔术扫描每个通道是否处于 High alarm 状态.

dwLowAlarm(bit 0~15)

Bit	F	E	D	C	B	A	9	8	7	6	5	4	3	2	1	0
channel	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

dwLowAlarm(bit 16~31)

Bit	1F	1E	1D	1C	1B	1A	19	18	17	16	15	14	13	12	11	10
channel	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16

wLowAlarm 是 32 个 bit 的参数直接对应到 32 个通道。

Ch ? = 0 → 无 low alarm

Ch ? = 1 → 有 low alarm

范例:

dwLowAlarm=0 → 所有通道正常，皆无 low alarm

dwLowAlarm=1 → 通道 0 是 low alarm, 其它皆正常。

dwLowAlarm=3 → 通道 0、1 是 low alarm, 其它皆正常。

dwHighAlarm(bit 0~15)

Bit	F	E	D	C	B	A	9	8	7	6	5	4	3	2	1	0
channel	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

dwHighAlarm(bit 16~31)

Bit	1F	1E	1D	1C	1B	1A	19	18	17	16	15	14	13	12	11	10
channel	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16

dwHighAlarm 是 32 个 bit 的参数直接对应到 32 个通道。

Ch ? = 0 → 无 high alarm

Ch ? = 1 → 有 high alarm

Example:

dwHighAlarm=0 →所有通道正常，皆无 high alarm

dwHighAlarm=4 → 通道 2 是 high alarm, 其它皆正常。

dwHighAlarm=7 → 通道 0、1、2 是 high alarm,其它皆正常。

- ## 返回值:

无返回值

- ## 范例程序：

DEMO11.C

2.6.10 P1202_StopMagicScan

停止魔术扫描的程序

- **语法：**

  void P1202_StopMagicScan(void);

- **参数：**

  无参数

- **返回值：**

  无返回值

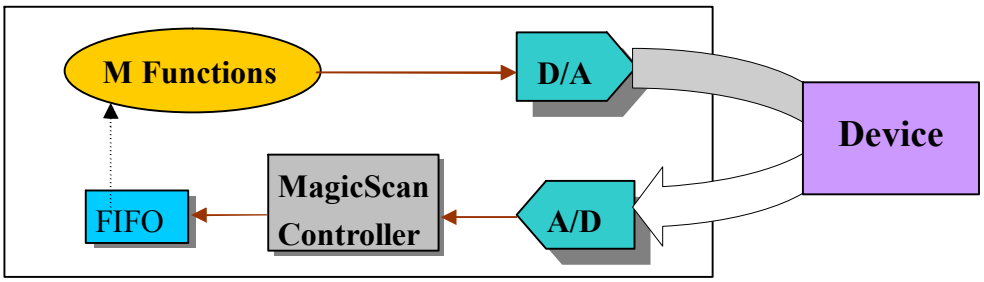
- **范例程序：**

  DEMO14.C

2.7 M_Functions 函数集

2.7.0 流程图

M_Function 应用程序运作原理：



※MagicScan Control：请参考第三章注 1。

表 2.9.0.1 M_Function 特性一览表

Function Name	D/A	A/D
M_FUN_1	D/A 通道 0(Analog Output 0) 输出正弦波	A/D 通道 0(Analog Input 0) +10V ~ -10V
M_FUN_2	D/A 通道 0(Analog Output 0) 输出值可自行设定	A/D 通道 0(Analog Input 0) +10V ~ -10V
M_FUN_3	D/A 通道 0(Analog Output 0) 输出正弦波	多个 A/D 通道 使用者决定(最多 32 个通道)
M_FUN_4	D/A 通道 0(Analog Output 0) 输出正弦波、方波、半方波	多个 A/D 通道 使用者决定(最多 32 个通道)

2.7.1 P1202_M_FUN_1

此函数是根据输入的点数与振幅，计算出描述一个正弦波的每个单点的数值，从板卡的 DA Channel 0 连续输出。使用者也可以设定要将这组正弦波数值连续多次输出，输出几组由使用者自行设定。输出的同时间从 AD Channel 0 读取资料，使用者可以设定要读取的速度与资料数。

参考「PCI-1202/1602/1800/1802 Hardware manual」第五章有更多详细的内容.

※ **此函数输出波形到 analog output 0 的同时，会从 analog input 0 读取资料。**

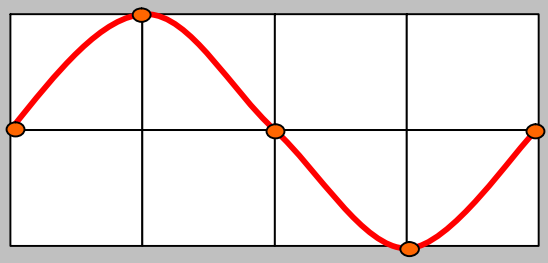
- **语法：**

WORD P1202_M_FUN_1(WORD wDaNumber, WORD wDaWave, float fDaAmplitude, WORD wAdSampleRateDiv, WORD wAdNumber, WORD wAdConfig, float fAdBuf[], float fLowAlarm, float fHighAlarm)

- **参数：**

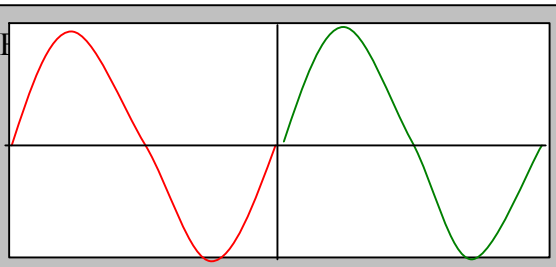
wDaNumber: [Input] 描述一组输出的正弦波要用的点数。点数越多，波形会越像真正的模拟输出；但因输出的点数多，输出一个完整波的速度会变慢。

Example1:



左图是将 wDaNumber 设为 5, 所以他将会输出一组有五个 DA 值的正弦波，而这五个值由程序自动计算而生成一个可以摸拟波型的五个点。要送出一组同样点数的波的时间通常取决于计算机的效能，效能越好时间越短。

wDaWave : [Input] 输出多少组的正弦波。



当 wDaWave 设定为 2 时。将会送出二组正弦波。

fDaAmplitude : [Input]正弦波的振幅.

> ※fDaAmplitude 范围为 -10~10
> 注意 : 板卡上的 J1 一定要设定成 +/-10V

wAdSampleRateDiv: [Input]**AD 取样频率 = 8M/ wAdSampleRateDiv** (取样/秒)
wAdNumber: [Input]读取 AD 资料的笔数。
wAdConfig : [Input]配置码，设定错误会量测到不正确的值请小心。

> **请参考 1.3 节**

fAdBuf[] : [Output] 为一浮点数数组。用来储存 A/D 资料的 buffer
fLowAlarm : [Input]最低警戒值. 当 **fAdBuf[?]**小于 **fLowAlarm** → LowAlarm
fHighAlarm : [Input]最高警戒值. 当 **fAdBuf[?]**大于 **fHighAlarm** → HighAlarm

- **返回值：**

  0 : 无错误
  其它 : 请参考 1.2 节 : 错误码定义。

- **范例程序：**DEMO5.C

2.7.2 P1202_M_FUN_2

其本上跟上一个函数的用法类似，只是本函数输出的波形是由使用者自由设定每一个 DA 值，而不像 P1202_M_FUN_1 是由函数自动产生。

参考「PCI-1202/1602/1800/1802 Hardware manual」第五章有更详细的内容。
※此函数输出波形到 analog output 0 的同时，会从 analog input 0 读取资料。

- **语法：**

WORD P1202_M_FUN_2(WORD wDaNumber, WORD wDaWave, WORD wDaBuf[], WORD wAdSampleRateDiv, WORD wAdNumber, WORD wAdConfig, WORD wAdBuf[]);

- **参数：**

wDaNumber: [Input] 描述使用者自由定义的波形使用的点数。

请参考 P1202_M_FUN_1 的范例 1

wDaWave : [Input] 输出多少组完整波形。

请参考 P1202_M_FUN_1 的范例 2

wDaBuf[] : [Output]储存使用者自由定义的波形的数值数组。
wAdSampleRateDiv: [Input]**AD 取样频率 = 8M/ wAdSampleRateDiv** (取样数/秒)
wAdNumber: [Input] 读取 AD 资料的笔数
wAdConfig : [Input] 配置码，设定错误会量测到不正确的值请小心。

请参考 1.3 节

wAdBuf[] : [Output] 为一浮点数数组。用来储存 A/D 资料的内存空间。

- **返回值：**

0 : 无错误
其它 : 请参考 1.2 节 : 错误码定义。

- **范例程序：**

DEMO7.C

2.7.3 P1202_M_FUN_3

此函数是根据输入的点数与振幅，计算出符合的正弦波的每个单点的数值，从板卡的 DA Channel 0 连续输出。使用者也可以设定要将这组正弦波数值连续多次输出，输出几组由使用者自行设定。输出的同时间从使用者选择的一个或多个输入通道读取资料，使用者可以设定要读取的速度与资料数。

参考「PCI-1202/1602/1800/1802 Hardware manual」第五章有更多详细的内容.
※此函数输出波形到 analog output 0 的同时，会从指定的通道读取数据。

- **语法：**

WORD P1202_M_FUN_3(WORD wDaNumber, WORD wDaWave, float fDaAmplitude, WORD wAdSampleRateDiv, WORD wAdNumber, WORD wChannelStatus[], WORD wAdConfig[], float fAdBuf[], float fLowAlarm, float fHighAlarm)

- **参数：**

wDaNumber: [Input] 描述一组输出的正弦波要用的数值点数。点数越多，波形会越像真正的模拟输出；但因输出的点数多，输出一个完整波的速度会变慢。

请参考 P1202_M_FUN_1 的范例 1

wDaWave : [Input] 输出多少组的正弦波。

请参考 P1202_M_FUN_1 的范例 2

fDaAmplitude : [Input] 正弦波的振幅.

※fDaAmplitude value range = -10~10 注：跳线 J1 需要选择成+/-10V

wAdSampleRateDiv: [Input]**AD 取样频率 = 8M/** wAdSampleRateDiv (取样数/秒)
wAdNumber: [Input] 读取 AD 资料的笔数
wAdChannel[]:[Input]数组的索引值对应到通道号码,1=要读取此通道资料,
0=不读取此通道数据

范例 3: wAdChannel**[0]** = 1 →要读取 Channel **0** 的资料 wAdChannel**[0]**=0 →不读取 Channel **0** 的资料 wAdChannel**[2]**=1 →要读取 Channel **2** 的资料

wAdConfig[]:[Input] **配置码，设定错误会量测到不正确的值请小心。**

请参考 1.3 节

fAdBuf[] : [Output] 为一浮点数数组。用来储存 A/D 的资料
fLowAlarm : [Input]最低警戒值. 当 **fAdBuf[?]小于 fLowAlarm** → LowAlarm
fHighAlarm : [Input]最高警戒值. 当 **fAdBuf[?]大于 fHighAlarm** → HighAlarm

- ## 返回值：

0 : 无错误
其它 : 请参考 1.2 节 : 错误码定义。

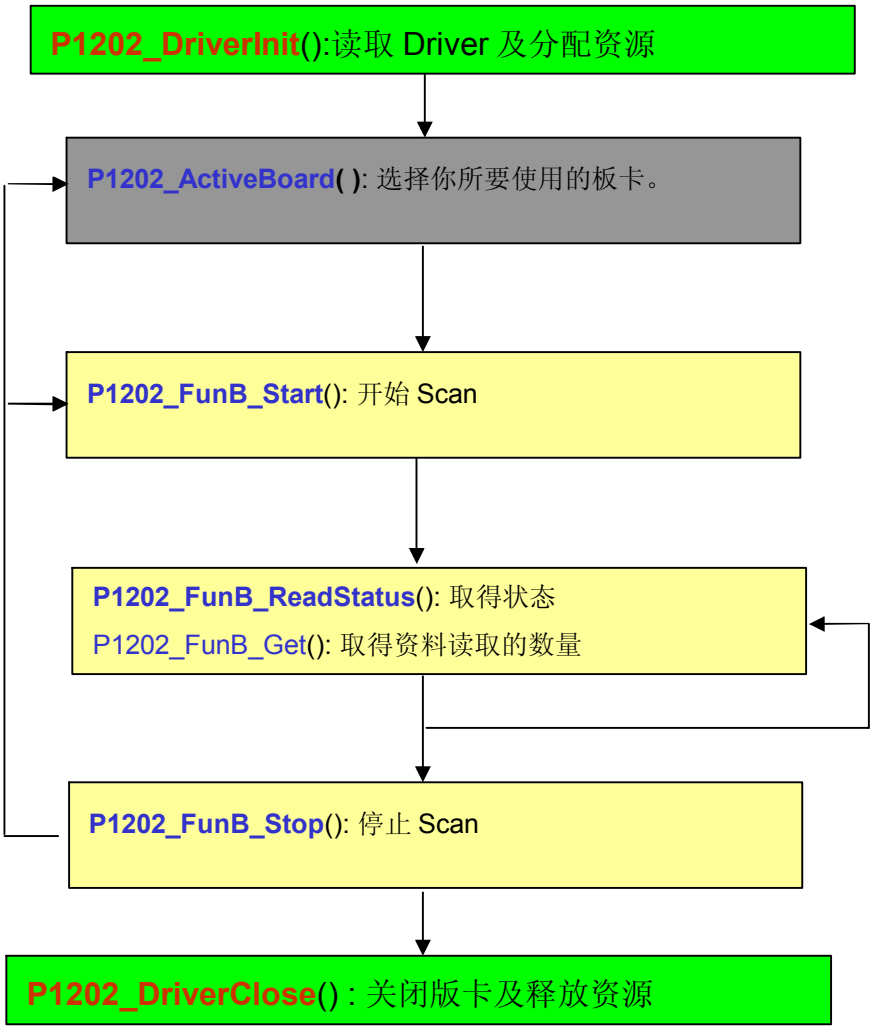
- ## 范例程序：

DEMO9.C

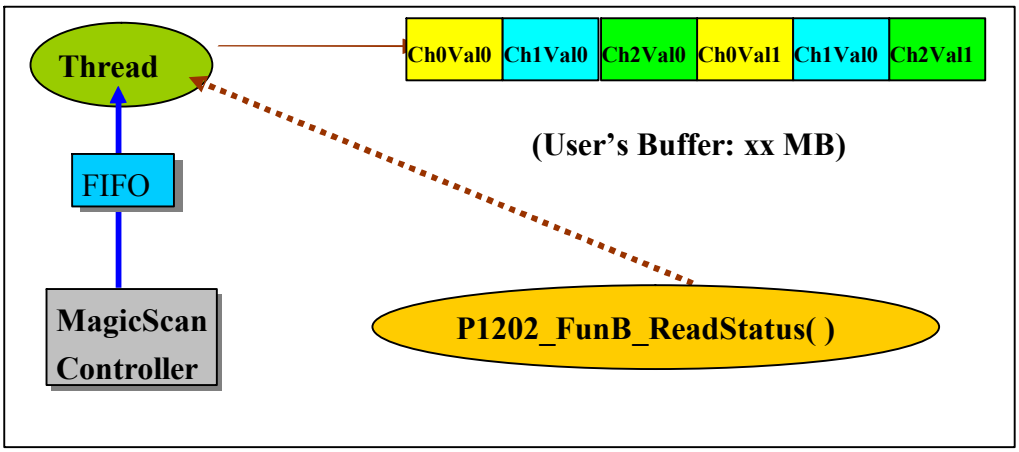
2.8 单个板卡批次采集

2.8.0 流程图

FunB 批次采集流程图



体系结构



2.8.1 P1202_FunB_Start

计算机上只安装一张板卡时，使用这个函数以连续的方式取值并将值储存在自行分配的内存里。详细的运作方式可以参考 2.9.0 节的说明。

- **语法：**

```
WORD P1202_FunB_Start(WORD wClockDiv0, WORD wChannel0[],
                      WORD wConfig0[], WORD *Buffer0, DWORD dwMaxCount0,
                      SHORT nPriority);
```

- **参数：**

wClockDiv0:[Input]取样频率。取样频率等于 8M/wClockDiv0.

wChannel0[]:[Input](0=no scan, 1=scan) 设定索引值对应的通道是否加入扫描取值。

> 范例 1:
> wChannel0**[0]** = 1 →Scan Channel **0 for first board**
> wChannel0**[0]**=0 →No scan Channel **0 for first board**
> wChannel0**[2]**=1 →Scan Channel **2 for first board**

wConfig0[]:[Input]配置码。请参考 1.3 节

*Buffer0: [Onput]储存读取资料的内存

dwMaxCount0: [Input] 该张板卡 buffer 的资料长度

nPriority :[Input] 设定线程的优先权(Priority)。请参考 1.4 节。

- **返回值：**

0 : 无错误

其它 : 请参考 1.2 节 : 错误码定义。

- **范例程序：**

DEMO21.C

2.8.2 P1202_FunB_ReadStatus

读取批次采集过程的状态.

- **语法：**

  WORD P1202_FunB_ReadStatus(void);

- **参数：**

  无参数

- **返回值：**

  0: 资料已准备好, 可以取回储存或分析使用

  1: 资料未准备好

- **范例程序：**

  DEMO21.C

2.8.3 P1202_FunB_Stop

停止批次采集过程.

- **语法:**

word P1202_FunB_Stop(void);

- **参数:**

无参数

- **返回值:**

0 : 无错误

其它 : 请参考 1.2 节 : 错误码定义。

- **范例程序：**DEMO21.C

2.8.4 P1202_FunB_Get

取得 FunB A/D 资料的读取资料笔数。

- **语法:**

word P1202_FunB_Get(DWORD *P0);

- **参数:**

*P0: [Output] A/D 资料的读取资料笔数。

- **返回值:**

0 : 无错误

其它 : 请参考 1.2 节 : 错误码定义。

- **范例程序：**DEMO21.C

2.9 多个板卡批次采集
(两个板卡同时工作)

2.9.0 P1202_FunA_Start

计算机上安装两张板卡时，使用这个函数可以一次规划两张板卡连续读取资料并将值储存在自行分配的内存里。详细的运作方式可以参考 2.9.0 节的说明。

- **语法：**

```
WORD P1202_FunA_Start(WORD wClockDiv0, WORD wChannel0[],
                      WORD wConfig0[], WORD *Buffer0, DWORD dwMaxCount0,
                      WORD wClockDiv1, WORD wChannel1[],
                      WORD wConfig1[], WORD *Buffer1, DWORD dwMaxCount1
                      Short  nPriority);
```

- **参数：**

wClockDiv0: [Input]第一张板卡的取样频率。取样频率是 8M/wClockDiv0。

wChannel0[]:[Input](0=no scan, 1=scan) 设定第一张板卡的通道是否加入扫描取值：数组的索引值对应到同号码的通道，1 为加入扫描取值，0 则不对此通道取值。

Example1: wChannel0**[0]** = 1 →Scan Channel **0 for first board** wChannel0**[0]**=0 →No scan Channel **0 for first board** wChannel0**[2]**=1 →Scan Channel **2 for first board**

wConfig0[]:[Input]设定第一张板卡的配置值。

请参考 1.3 节

*Buffer0: [Output]储存第一张板卡的 A/D 资料的 buffer。

dwMaxCount0:[Input] 第一张板卡 buffer 的资料长度

wClockDiv1: [Input] 第二张板卡的取样频率。取样频率是 8M/wClockDiv0。

wChannel1[]:[Input] (0=no scan, 1=scan) 设定第一张板卡的通道是否加入扫描取值：数组的索引值对应到同号码的通道，1 为加入扫描取值，0 则不对此通道取值。

范例 1:
wChannel1**[0]** = 1 →Scan Channel **0 for second board**
wChannel1**[0]**=0 →No scan Channel **0 for second board**
wChannel1**[2]**=1 →Scan Channel **2 for second board**

wConfig1[]:设定第二张板卡的配置值。

请参考 1.3 节

*Buffer1: [Output] 储存第二张板卡的 A/D 资料的 buffer。
dwMaxCount1: [Input] 第二张板卡 buffer 的资料长度
nPriority:[Input] 设定线程的优先权(Priority)。请参考 1.4 节。

- **返回值：**

0 : 无错误
其它 : 请参考 1.2 节 : 错误码定义。

- **范例程序：**DEMO20.C

2.9.1 P1202_FunA_ReadStatus

用来取得批次采集过程的状态.

- **语法：**

WORD P1202_FunA_ReadStatus(void);

- **参数：**

无参数

- **返回值:**

0: 资料已准备好, 可以取回储存或分析使用
1: 资料未准备好。

- **范例程序：**DEMO20.C

2.9.2 P1202_FunA_Stop

停止批次采集函数.

- **语法:**

word P1202_FunA_Stop(void);

- **参数:**

无参数

- **返回值:**

无返回值

- **范例程序：**

DEMO20.C

2.9.3 P1202_FunA_Get

取得 FunA A/D 资料的读取笔数。

- **语法:**

word P1202_FunA_Get(DWORD *P0, DWORD *P1);

- **参数:**

*P0:[Output]第一张板卡有多少笔 A/D 资料被读取。

*P1:[Output] 第二张板卡有多少笔 A/D 资料被读取。

- **返回值:**

0 : 无错误

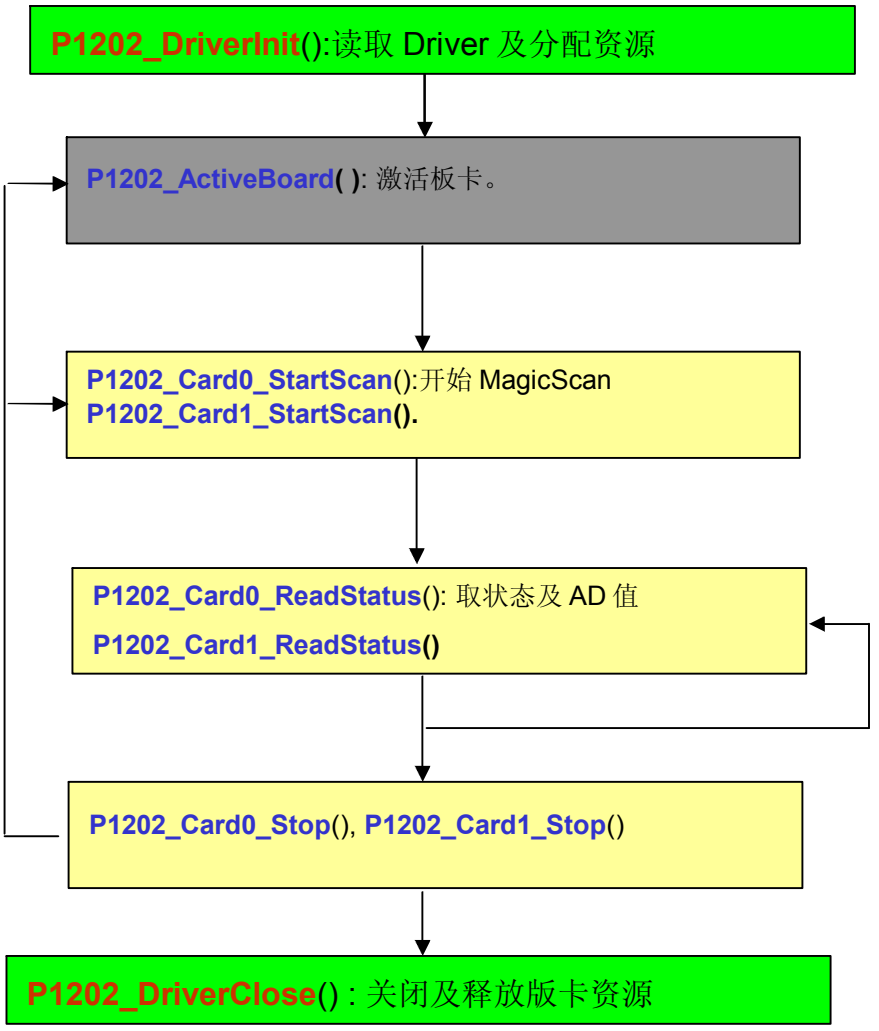
其它 : 请参考 1.2 节 : 错误码定义。

- **范例程序：**DEMO20.C

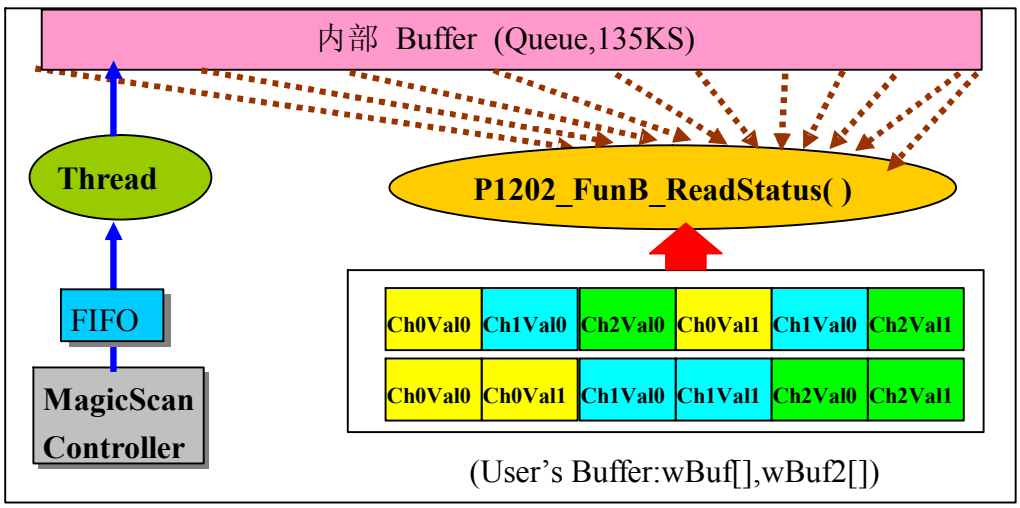
2.10 连续采集函数集

2.10.0 流程图

连续采集流程图



体系结构



2.10.1 P1202_Card0_StartScan

这个函数会使用第一张卡以扫描通道，连续取值的方式读取 AD 值。建议使用在较低的速度及较长时间来使用。否则内部 buffer 溢满的机会将会增高。依各人的计算机状况不同，建议取样频率最好在 40kHz 以下。请参考「 PCI-1202/1602/180x Hardware User Manual」可以得到其它的参考信息。

- **语法：**

WORD P1202_Card0_StartScan(WORD wSampleRateDiv, WORD wChannelStatus[], WORD wChanelConfig[], WORD wCycles);

- **参数：**

wSampleRateDiv :[Input] **AD 取样频率 = 8M/wSampleRateDiv.**

wSampleRateDiv=80 → 取样频率=8M/80=100K Hz

wChannelStatus[] :[Input] (0=no scan, 1=scan) 设定每一个通道是否加入扫描取值：数组的索引值对应到同号码的通道，1 为加入扫描取值，0 则不对此通道取值。

Example1: wChannelStatus **[0]** = 1 →Scan Channel **0** wChannelStatus **[0]**=0 →No scan Channel **0** wChannelStatus **[2]**=1 →Scan Channel **2**

wChannelConfig[] :[Input] 设定每个通道的配置值，数组的索引值对应到同号码的通道。例如 wChannelConfig[1]=0←设定通道 1 的配置值为 0。

wCycles:[Input] 因为此函数是连续不停的读取资料，由函数分次将资料读入，所以需使用此参数设定一次数据读入时要取得每一个通道上的数据数。这个次数没有限制。 一次取回的数据量=wCycles * 加入扫描的通道数

- **返回值：**

0 : 无错误

其它 : 请参考 1.2 节 : 错误码定义。

- **范例程序：**DEMO13.C

2.10.2 P1202_Card0_ReadStatus

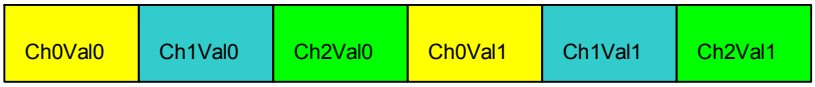
这个函数用来取得连续读取程序的状态及检查资料搜集是否完成。在连续取值的过程中，不会阻止程序的运作。您必须使用这个返回值，若为 0 表示资料是准备好的，你可以取回资料并继续检查程序的状态，再取回下一次搜集完的资料，直到长时间连续取值工作完成为止。

- **语法：**

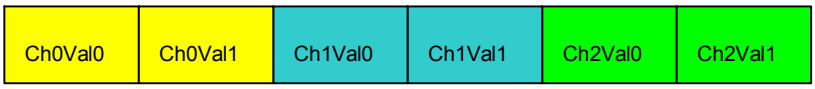
P1202_Card0_ReadStatus(WORD wBuf[], WORD wBuf2[], DWORD *dwP1, DWORD *dwP2, WORD *wStatus);

- **参数：**[output]

wBuf[] :[Output]储存数据数组，依照扫描顺序储存数据。
扫描顺序为(012…N012…N……012…N)的话，读取资料储存顺序如



wBuf2[] :[Output] 储存数据数组。依照通道别将数据储存如
(0000…..11111……22222….NNNNN….)



*dwP1 :[Input/Output] 保留给内部使用.
*dwP2 :[Input/Output] 保留给内部使用.
*wStatus :[Output] 1=开始线程, 2=超过时间, 8=FIFO 溢满, 0x80=结束线程, 5=Buffer 已满

- **返回值：**

0: 资料已准备好
1: 无资料
2.:FIFO 溢满
3: 线程超过时间
4: Buffer 已满

- **范例程序：**DEMO13.C

2.10.3 P1202_Card0_Stop

停止连续读取资料的程序。

- **语法：**

  void P1202_Card0_Stop(void);

- **参数：**

  (None)

- **返回值:**

  0 : 无错误

  其它 : 请参考 1.2 节 : 错误码定义。

- **范例程序：**DEMO13.C

2.10.4 P1202_Card1_StartScan

这个函数会使用第二张卡以连续取值的方式读取 AD 值。建议使用在较低的速度及较长时间来使用。否则内部 buffer 溢满的机会将会增高。依各人的计算机状况不同，建议取样频率最好在 40kHz 以下。请参考「 PCI-1202/1602/180x Hardware User Manual」可以得到其它的参考信息。

- **语法：**

  WORD P1202_Card1_StartScan(WORD wSampleRateDiv, WORD wChannelStatus[], WORD wChanelConfig[], WORD wCount);

- **参数：**

  wSampleRateDiv :[Input] **AD 取样频率 = 8M/wSampleRateDiv.**

  wSampleRateDiv=80 → 取样频率=8M/80=100K Hz

  wChannelStatus[] :[Input] (0=no scan, 1=scan) (0=no scan, 1=scan) 设定每一个通道是否加入扫描取值：数组的索引值对应到同号码的通道，1 为加入扫描取值，0 则不对此通道取值。

  范例 1:
  wChannelStatus**[0]** = 1 →Scan Channel **0**
  wChannelStatus**[0]**=0 →No scan Channel **0**
  wChannelStatus**[2]**=1 →Scan Channel **2**

  wChanelConfig[] :[Input] 设定每个通道的配置值(1.3 节)，数组的索引值对应到同号码的通道。例如 wChannelConfig[1]=0←设定通道 1 的配置值为 0。

  wCycles:[Input] 因为此函数是连续不停的读取资料，由函数分次将资料读入，所以需使用此参数设定一次数据读入时要取得每一个通道上的数据数。这个次数没有限制。 一次取回的数据量=wCycles * 加入扫描的通道数

- **返回值：**

  0 : 无错误

  其它 : 请参考 1.2 节 : 错误码定义。

- **范例程序：**DEMO14.C

2.10.5 P1202_Card1_ReadStatus

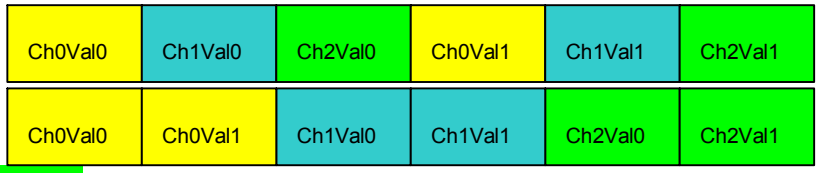
这个函数用来取得连续读取程序的状态及检查资料搜集是否完成。在连续取值的过程中，不会阻止程序的运作。您必须使用这个返回值，若为 0 表示资料是准备好的，你可以取回资料并继续检查程序的状态，再取回下一次搜集完的资料，直到长时间连续取值工作完成为止。

- 语法：

P1202_Card1_ReadStatus(WORD wBuf[], WORD wBuf2[], DWORD *dwP1, DWORD *dwP2, WORD *wStatus);

- 参数：[output]

wBuf[] :[Output]数组。用来存入扫描资料，顺序为(012…N012…N……012…N)



wBuf2[] :[Output]数组。用来存入扫描资料，顺序为(0000…..11111……22222….NNNNN….)

*dwP1 :[Input/Output] 保留给内部使用.

*dwP2 :[Input/Output] 保留给内部使用.

*wStatus :[Output] 1=开始线程, 2=超过时间, 8=FIFO 溢满, 0x80=结束线程, 5=Buffer 已满

- 返回值:

0: 资料已准备好

1: 无资料

2.:FIFO 溢满

3: 线程超过时间

4: Buffer 已满

- 范例程序：DEMO14.C

2.10.6 P1202_Card1_Stop

停止连续读取的程序

- **语法：**

  void P1202_Card1_Stop(void);

- **参数：**

  无参数

- **返回值：**

  无返回值

- **范例程序：**

  DEMO14.C

2.11 其它函数

2.11.0 P1202_DelayUs

这个函数驱动板卡上的一个独立的定时器，用来计算一小段时间或延迟一小段时间. 它只能使用在激活状态中的板卡，请使用 P1202_ActiveBoard(2.2.4 节) 激活您要使用的板卡。

- 语法：

  word P1202_DelayUs(WORD wDelayUs);

- 参数：

  wDelayUs : 延迟时间(μs),最大值为 8191 μs

  wDelayUs=1 → 延迟 1 us

  wDelayUs=1000 → 延迟 1000 us = 1 ms

  wDelayUs=8191 → 延迟 8191 us = 8.191 ms (最大的延迟时间)

  wDelayUs=8192 → 无效 delay (将会回传错误码)

- 返回值：

  0 : 无错误

  其它 : 请参考 1.2 节 : 错误码定义。

- 范例程序：DEMO1.C

- **Long Time Delay :**

```
WORD DelayMs(WORD wDelayMs) // maximum delay=4294967.295 sec
{
WORD wDelay,wRetVal

wRetVal=0;
for  (wDelay=0; wDelay<wDelayMs; wDelay++)
     wRetVal+=P1202_DelayUs(1000);
return(wRetVal);
}
```

3. 附录

※ 注 1:

[MagicScan controller]
这是一种硬件扫描的机制，程序设定扫描的信道顺序与数入范围的配置值，之后激活硬件扫描程序。一个硬件的 **MagicScan Controller** 就会依照指定的频率, 顺序与输入信号范围配置值，在指定通道间切换，取值，并将取得的 **AD** 值放到 **FIFO**，**driver** 同时将 **FIFO** 的资料搬到 **Buffer** 中，使用者的程序只须使用函式将 **Buffer** 内的资料取出。这样的处理方式可以减少 **CPU** 的处理时间。

※ 注 2:

[FIFO Overflow]
当 **FIFO** 空间已经全满时，下一笔进入的资料将覆盖第一笔资料而造成资料遗失称为 **FIFO Overflow**。
要减少 **FIFO Overflow** 的发生，建议停止使用其它的程序像是 IE 或是防毒软件；或是选择 **FIFO** 容量更大的板卡像是 **8K** 的版本，通常这样可以有效降低 **FIFO Overflow** 发生的机率

4. 范例程序

PCI-1202 提供 20 个以上的范例程序, 每个范例功能条列如下:

- demo1: one board, D/I/O test, D/A test, A/D polling & pacer trigger test, general test
- demo2: two board, same as demo1
- demo3: one board, all 32 channels of A/D by software trigger(by polling)
- demo4: two board, same as demo3
- demo5: one board, M_function_1 demo
- demo6: two board, same as demo5
- demo7: one board, M_function_2 demo
- demo8: two board, same as demo7
- demo9: one board, M_function_3 demo
- demo10: two board, same as demo9
- demo11: one board, MagicScan demo
- demo12: two board, same as demo11
- demo13: one board, continuous capture demo
- demo14: two board, continuous capture demo (Windows 95/NT only)
- demo15: all installed board, D/I/O test for board number identification
- demo16: one board, performance evaluation demo
- demo17: one board, MagicScan demo, scan sequence: 4→3→5
- demo18: one board, MagicScan demo, scan 32 channel, show channel 0/1/15/16/17
- demo19: one board, A/D calibration.
- demo20: two board, P1202_FUNA, continuous capture demo
- demo21: single board, P1202_FUNB, continuous capture demo
- demo23: single board, post-trigger demo
- demo24: single board, pre-trigger demo
- demo25: single board, middle-trigger demo

详细资料参考配送光盘。