

**Адаптер
аналогового
и дискретного ввода-вывода**

IC540IO

Описание программного интерфейса

Содержание

1. Назначение	
2. Драйвер адаптера аналогового ввода IC-540AI	
2.1. Типы данных	
2.2. Константы	
2.3. Переменные	
2.4. Процедуры	
3. Пример программирования	
Приложение. О регистрах ПДП	

1. Назначение

Описание содержит достаточную информацию по программированию адаптера аналогового ввода IC-540AI и состоит из 2 частей:

- описание содержимого драйвера
- основы написания программ

2. Драйвер адаптера аналогового ввода IC-540AI

Драйвер написан на языке Pascal с использованием встроенного Ассемблера 286. Применение данной программы, также как и адаптера, на менее мощных компьютерах невозможно. Драйвер содержит в себе все необходимые для работы с адаптером константы, переменные, функции. Если Вам покажется недостаточным объем информации - рекомендуем обратиться к исходному тексту драйвера.

2.1 Типы данных

DataType - квант данных выходного буфера

Имеет размер 4 байта. В техническом описании и процедуре TInterior.Data2DataText программы BUFMAKER показано, как данные этого типа преобразуются в сигналы управления адаптером.

DataTextType = преобразованный ("осмысленный") квант данных выходного буфера

DataTextType = record

Time : byte; {время цикла в тиках (около 12мкс)}

Irq : IrqType; {прерывание}

Mult : byte; {регистр управления мультиплексором}

case Work : WorkType of {работа - настройка}

Wrk:

(Scal : ScalType; {шкала}

Cent : CentType; {позиция центра}

Sch : SchType; {схема подключения});

Stp:

SchSr : SchSrType; {схема подключения источника}

SchAd : SchAdType; {схема подключения АЦП})

Для вывода информации об этих данных на экран в программе BUFMAKER использована процедура TInterior.DataText2String, для преобразования в квант данных выходного буфера - процедура TInterior.DataText2Data.

Для облегчения восприятия большинство полей в типе DataTextType заданы перечислимым типом:

WorkType = (Wrk,Stp);

IrqType = (IYes,INo);

ScalType = (Scal0, Scal1, Scal2, Scal3, Scal4, Scal5, Scal6, Scal7);

CentType = (Cent0, Cent1, Cent2, Cent3);

SchType = (Sch0, Sch1, Sch2, Sch3, Sch4, Sch5, Sch6, Sch7);

NomSrType = (NS0, NS1, NS2, NS3);

SchSrType = (SS0, SS1, SS2, SS3);

SchAdType = (SA0, SA1, SA2, SA3);

Также перечислимым типом описана переменная режима инициализации ПДП

InitModeType=(MMono,MMult);

2.2 Константы

В модуле определены адреса основных регистров контроллера ПДП:

DMA_WRR, DMA_Mask, DMA_WMR, DMA_FF, DMA_WMC, DMA_RTR, DMA_CMР,
DMA_AMR, Dma_Adr5, Dma_Adr6, Dma_Adr7, Dma_Pag5, Dma_Pag6, Dma_Pag7

2.3 Переменные

IRQ:byte - номер используемого прерывания (допустимы 10,11,12,15)

DRQIN:byte - номер канала ПДП на прием (5,6)

DRQOUT:byte - номер канала ПДП на передачу (6,7)

IODELAY:word - адрес порта для формирования задержки

CalibrR:real - значение калибровочного резистора

Следующие 6 переменных указывают адреса регистров ПДП, зависящих от номера используемых каналов ПДП

DMAi_Adr:word - базовый адрес канала ввода данных

DMAi_Cou:word - счетчик

DMAi_Pag:word - страничный регистр

DMAo_Adr:word - базовый адрес канала вывода данных

DMAo_Cou:word - счетчик

DMAo_Pag:word - страничный регистр

InitMode:InitModeType - режим инициализации ПДП. Может принимать значение MMono (однократный проход буферов приема и передачи и останов адаптера) и MMult (непрерывная работа адаптера по приему данных)

AdcPresent:boolean - в этой переменной возвращается результат проверки наличия адаптера. Кроме того она управляет работой программы инициализации, запрещая запускать ПДП если AdcPresent:=false.

Для программ, использующих прерывание от адаптера зарезервированы 2 переменные

IntOldAddr:pointer - адрес старого обработчика IRQ и

IntOldMask:byte - старая маска контроллера прерываний

BufInP:pointer - указатель на входной буфер

BufOutP:pointer - указатель на выходной буфер

BufInSize:word - размер буфера ввода

BufOutSize:word - размер буфера вывода

Четыре вышеописанных переменных являются основными для большинства процедур драйвера. Они задают адреса буферов в формате СЕГМЕНТ:СМЕЩЕНИЕ и их размер в 16ти разрядных словах

BufIn:word - начальный адрес буфера ввода

BufInSeg:word - страничный регистр буфера ввода

BufOut:word - начальный адрес буфера вывода

BufOutSeg:word - страничный регистр буфера вывода

Эти переменные предназначены для ускорения процедуры инициализации. Они содержат числа, заносимые в регистр стартового адреса контроллера ПДП и страничный регистр. При этом учитывается, что каналы ПДП 5,6,7, задействованные в адаптере, не могут обращаться к адресам с ненулевыми A24..A31. Начальный адрес буфера обязан иметь A0=0. Таким образом в переменных Buf?? хранится значение, записываемое в регистр базового адреса ПДП т.е. биты A1..A16 полного адреса. В переменных Buf??Seg хранится значение 8-ми разрядного страничного регистра (младший бит страничного регистра у DMA 5,6,7 игнорируется), т.е. биты A16..A23 полного адреса.

При инициализации или останове в момент перекачки данных адаптера для осуществления гарантированной синхронизации схемы необходимо произвести однократный его запуск. Для осуществления этой процедуры введен специальный буфер с параметрами:

BufInitIn:word - начальный адрес буфера ввода при инициализации

BufInitInSeg:word - страничный регистр буфера ввода -||-

BufInitOut:word - начальный адрес буфера вывода -||-
BufInitOutSeg:word - страничный регистр буфера вывода -||-

2.4 Процедуры

procedure SetConf (_DrqIn,_DrqOut,_Irq:byte;_IoDelay:word);

Процедура устанавливает все необходимые переменные, однозначно определяющие адресацию адаптера (номера линий ПДП и прерываний), а также адрес устройства ввода/вывода для формирования гарантированной задержки. Обычно процедура вызывается из FileLoadCLB. Однако, если по каким-либо причинам FileLoadCLB не используется (например в тестовых программах, а также в случаях, когда надо следить не за абсолютными параметрами, а за их соотношением и изменением), процедура должна вызываться один раз в начале программы, до вызова любых других процедур данного модуля.

_DrqIn - номер канала ПДП, передающего принятые данные от адаптера (допустимые значения - 5, 6). По умолчанию 5.

_DrqOut - номер канала ПДП, передающего управляющие данные в адаптер (допустимые значения - 6, 7). По умолчанию 6.

_Irq - номер прерывания (допустимые значения - 10, 11, 12,15). По умолчанию 15.

_IoDelay - адрес устройства ввода-вывода для формирования гарантированной задержки. По умолчанию 200h. По заданному адресу может (и даже это рекомендуется) не находится никакого устройства. Нежелательно в качестве такого адреса выбирать адрес устройства, расположенного на процессорной плате.

function SetBuf : word;

Функция определяет все внутренние переменные, требуемые для инициализации (запуска) работы адаптера. Функция использует указатели на входной и выходной буферы BufInP и BufOutP, которые должны быть установлены до вызова функции. Согласно логике работы адаптера, начало буферов должно быть выровнено по границе слова (A0=0). Функция осуществляет это выравнивание. Сдвиг входного буфера заносится в младший байт результата, сдвиг выходного - в старший. В связи с таким способом выравнивания рекомендуется всегда задавать размер буфера хотя бы на 1 байт больше реально используемого, или выравнивать буфера самостоятельно до вызова функции SetBuf.

function FileLoad540(FN:string;var Size:word):boolean;

Файл параметров с именем FN загружается в область, указанную переменной BufOutP. При этом размер буфера (в словах) присваивается переменной Size. При неудачной загрузке функция принимает значение False.

procedure Init;

Основная процедура драйвера. Запускает адаптер с заданными предварительно параметрами.

procedure Done;

Останавливает работу адаптера. Маскирует задействованные адаптером каналы ПДП.

function HardDone:boolean;

Проверка наличия адаптера

procedure FullStartDMA;

Последовательно запускает процедуры SetBuf, Done, Init.

function DataRead1(Num:word):word;

Считывает принятый адаптером сигнал, соответствующий строке Num буфера вывода. Перед считыванием проверяет готовность данных.

Внимание! Функция может привести к заикливанию программы, например при остановке ПДП или если Num превышает реальные размеры буфера данных.

```
function DataRead2(Num:word):word;
```

Функция аналогична предыдущей, но без ожидания готовности (поэтому свободна от зависаний)

```
function DMAend:boolean;
```

Функция проверяет окончание цикла приема данных

3. Пример программирования

Обычно программирование адаптера осуществляется по следующей схеме

- Если программа при работе с драйвером не меняет динамические параметры выходного буфера и считывает принятые данные с помощью функций драйвера типа DataRead, то Вам нет необходимости объявлять какие-либо дополнительные переменные. В некоторых случаях можно объявить свои переменные, например для повышения удобства доступа ко входным и выходным данным. Обычно это массив 2байтовых слов для входного буфера и 4байтовых записей (тип DataaType) - для выходного.

Размер буферов определяется Вами. Помните однако, что необходимо чтобы буфер был выровнен по границе слова. Переменные драйвера BufInP и BufOutP типа Pointer предназначены для указания на Ваши буферы. Во многих случаях удобно задать буферы так как показано ниже.

- Не забывайте установить переменные BufInSize и BufOutSize - реальные размеры буферов в 16-разрядных (2-байтных) словах. Реальный размер - это объем памяти, необходимый для размещения выходного буфера (объем загруженного файла параметров *.540 в байтах разделить на 2) и входного буфера (еще в 2 раза меньше).

```
type
```

```
  TArO = array[0..1000] of DataType;
```

```
  PArO = ^TArO;
```

```
  TArI = array[0..1000] of word;
```

```
  PArI = ^TArI;
```

```
var
```

```
  ArI : PArI absolute BufInP;
```

```
  ArO : PArO absolute BufOutP;
```

- Устанавливаются номера задействованных линий ПДП и прерываний.

```
If not FileLoadCLB then Halt;
```

- Выделение области ОЗУ для выходного и входного буферов, с проверкой на пересечение границы 128*N кб и выравниванием по четному адресу

```
var
```

```
  BuferDelta:word;
```

```
  P:array[1..30]of PArI;
```

```
  K,L:byte;
```

```
begin
```

```
  K=0; {проверка на переход границы}
```

```
  repeat
```

```
    inc(K);
```

```
    if (K>30) then halt;
```

```
    new(P[K]);
```

```
  until (seg(P[K]^)div $2000)=((seg(P[P[K]^)+(SizeOf(P[K]^)div 16)) div $2000);
```

```

for L:=1 to K-1 do
  Dispose(P[L]);
ArI:=P[K];
end;

```

- Загружается файл параметров *.540. Устанавливаются реальные размеры буферов приема и передачи

```

if not FileLoad540('Example1.540', BufInSize) then Halt;
BufOutSize:=BufInSize*2;

```
- Проверяется наличие адаптера

```

if not HardDone then Halt

```
- Установка параметров инициализации ПДП

```

InitMode:=MMono;

```
- Запуск ПДП

```

Init;

```
- По готовности - чтение данных

```

repeat
until DmaEnd;
for K:=1 to BufInSize do
D:=DataRead1(K)

```
- Если установлен режим MMono, то после обработки данных повторяются два предыдущих пункта, в режиме MMult - один.
- При выходе из программы - маскирование ПДП

```

Done

```

Приложение. О регистрах ПДП.

Если Вы хотите попробовать свои силы в программировании на самом нижнем уровне (чего мы не рекомендуем, по крайней мере в первое время знакомства с адаптером) прочтите эту главу. Для программирования контроллера ПДП используются нижеописанные регистры (указаны только необходимые коды, подробные инструкции смотрите в специальной литературе. Из доступной в России можем рекомендовать например книгу 'Микропроцессорный комплект К1810' под редакцией Ю.М. Казаринова издательства 'Высшая школа' 1990).

Регистр установки режима WMR (write mask registr) - 0D6h

D7,D6	00	передача по требованию
	01	одиночная передача
	10	блочная передача
D5	0	приращение адреса - инкрементирование
	1	прорашение адреса - декрементирование
D4	0	нет автоинициализации
	1	автоинициализация
D3,D2	01	цикл записи в память
	10	цикл чтения из памяти
D1,D0	01	канал 5

10 канал 6
11 канал 7

Регистр слова-состояния RSR (read state register) - 0D0h

D7..D5 разрешение на ПДП каналов 7..5

D3..D1 сигнал ТС каналов 7..5

Регистр первый/второй FFP (flip/flop clear byte pointer) - 0D8h

Запись в этот регистр любого байта устанавливает указатель на первый байт

Регистр запросов WRR (write request register) - 0D2h

D2 0 сбросить маску
1 установить маску
D1,D0 01 канал 5
10 канал 6
11 канал 7

Дополнительно к этим регистрам каждый канал имеет свои собственные 16-разрядные регистры.

* Регистр текущего адреса CAR(current address register) - хранит текущий адрес ячейки памяти при выполнении цикла ПДП. Адреса для 5,6,7 каналов - 0C4h, 0C8h, 0CCh соответственно.

* Регистр циклов ПДП CWR() - хранит число слов для передачи. Помните, что загружаемая в него константа должна быть на 1 меньше числа слов для передачи. Адреса регистров 0C6h, 0CAh, 0CEh соответственно.

* Кроме того, для полной адресации используется так называемый страничный регистр, выводящий старшие разряды адреса ОЗУ. Для каналов 5,6,7 это будут регистры по адресам 08Bh, 089h, 08Ah.