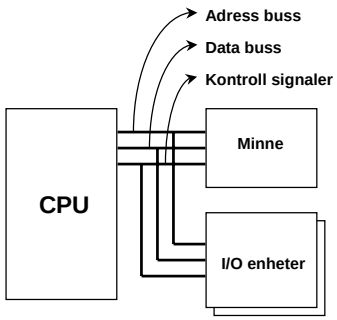
F6: I/O hantering

- **Typer av I/O i ett datorsystem**
  - **Memory mapped**
  - **Port mapped**
- **Protokoll för synkronisering**
  - **Polling**
  - **Timed**
  - **Interrupt**
  - **DMA**
- **Drivrutiner**

Mittuniversitetet
MID SWEDEN UNIVERSITY

Memory mapped

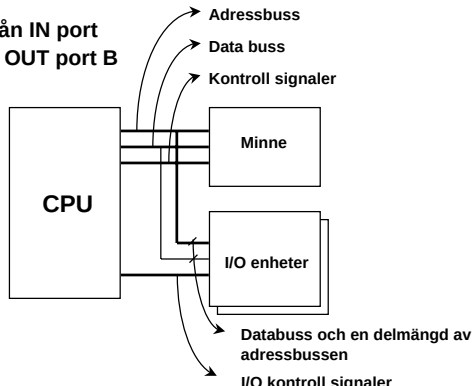
- **I/O enheter avkodas precis som ett minne och I/O befinner sig därför i samma adressrymd som minnet.**
- **Data accessas med vanliga minnesinstruktioner**
  - **Ex. MOVE.W $400700,D0 * Hämtar info från In porten som har adress $400700**



Mittuniversitetet
MID SWEDEN UNIVERSITY

Port mapped

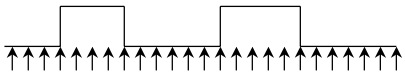
- Eget minnesutrymme, dvs en egen adressrymd
- Speciella instruktioner för åtkomst av I/O enheter
- Ex.
  - IN A * Läs in data från IN port
  - OUT B * Skriv data till OUT port B



Mittuniversitetet
MID SWEDEN UNIVERSITY

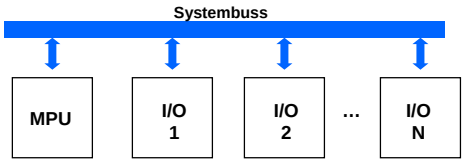
Synkronisering av I/O – Polling 1(2)

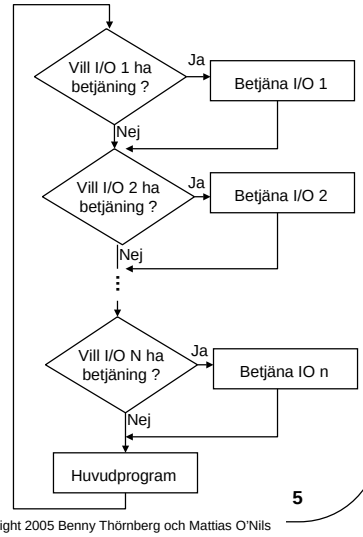
- Polling
  - I/O porten läses av med ett bestämt intervall
  - Kallas också för programmerad I/O
  - Sker synkront med övrigt programflöde



Mittuniversitetet
MID SWEDEN UNIVERSITY

Synkronisering av I/O – Polling 2(2)



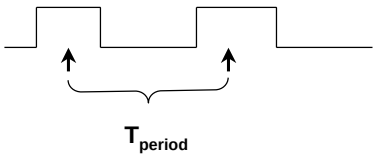


- **Polling av flera I/O**
  - **Varje I/O-enhet tillfrågas enligt en förutbestämd ordning om de behöver betjäning.**
  - **Om betjäning behövs körs det serviceprogram som är associerat med tillfrågad enhet.**
  - **Övrig tid ägnar processorn till att exekvera huvudprogrammet.**

Mittuniversitetet
MID SWEDEN UNIVERSITY

Synkronisering av I/O - Timed

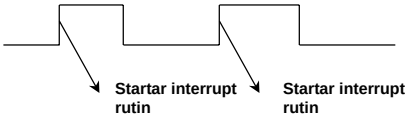
- **Timed**
  - **Intervallet mellan händelser är känt**
  - **Accessar I/O enheten efter en viss tid när man vet att en händelse har inträffat**
  - **Kräver realtids OS**



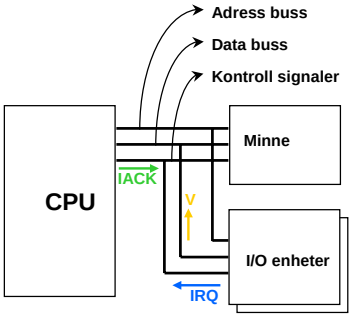
Mittuniversitetet
MID SWEDEN UNIVERSITY

Synkronisering av I/O – Avbrott 1(2)

- **Avbrott (Interrupt)**
  - **En händelse på in/ut-porten som behöver underhållas, avbryter då ordinarie programflöde och exekverar en speciell service rutin genom att någon interrupt-ingång på processorn aktiveras**
  - **Avbrottsrutinen som betjänar en I/O-enhet exekverar asynkront i relation till huvudprogrammet.**



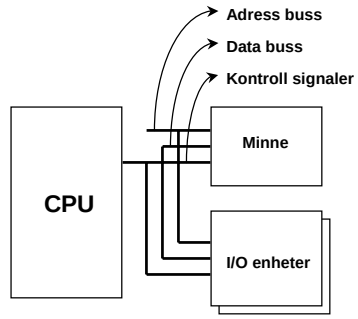
Synkronisering av I/O – Avbrott 2(2)



- **I/O-enheten signalerar till processorn att den behöver hjälp genom signalen Interrupt Request.**
- **Processorn avbryter det den för stunden arbetar med och signalerar samtidigt till I/O-enheten att den accepterar avbrottet – Interrupt Acknowledge.**
- **I/O-enheten identifierar sig nu genom att skicka en Vektor på databussen.**
- **Processorn servar I/O-enheten genom att utföra dataöverföringen till/från enheten.**
- **Processorn återgår till tidigare arbete.**

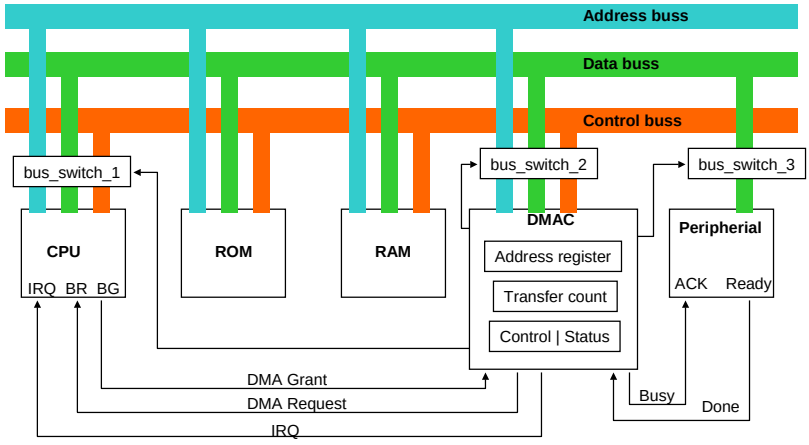
Synkronisering av I/O – DMA 1(2)

- **DMA – Direct Memory Access**
  - **Processorn är bortkopplad**
  - **I/O enheten får ett adressområde tilldelat**
  - **Det går snabbare att utföra blocköverföring av data**
  - **Överföringen sker snabbt i en skur eller blandas med processorns busscykler**



Mittuniversitetet
MID SWEDEN UNIVERSITY

Synkronisering av I/O – DMA 2(2)



Genom att öppna bus_switch_1 och samtidigt sluta 2 och 3 så kan DMAC kontrollera en dataöverföring till/från minnet.

Mittuniversitetet
MID SWEDEN UNIVERSITY

Sammanfattning av I/O 1(2)

- **Memory mapped**
  - **Varje I/O port har en minnesadress i den gemensamma minnesarean**
- **Port mapped**
  - **I/O portarna har ett eget adressutrymme skilt från den vanliga minnesarean. Speciella I/O-instruktioner.**

Sammanfattning av I/O 2(2)

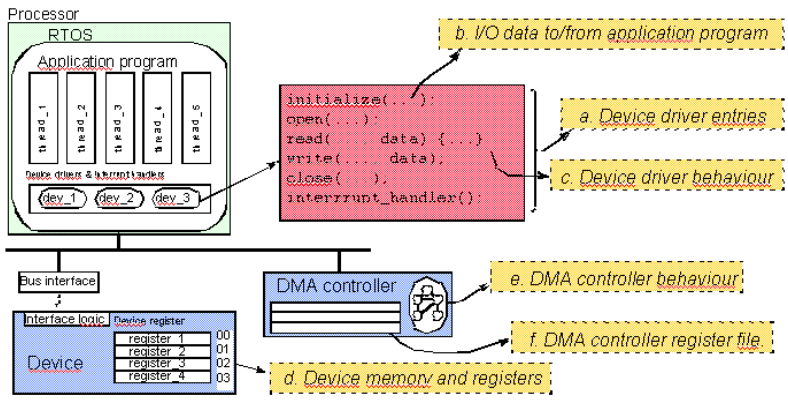
- **Polling eller programmerad I/O**
  - **I/O porten läses av med ett bestämt intervall för att undersöka i fall en viss händelse inträffat.**
- **Timed I/O**
  - **Periodtiden mellan varje händelse är känd.**
- **Avbrottsbaserad I/O**
  - **I/O-enheten signalerar själv till processorn när en händelse inträffat och en dataöverföring behöver ske.**
- **DMA**
  - **Direct Memory Access**
  - **Processorn kopplas bort. I/O enheten läser och skriver i minnet utan att data passerar genom processorn**

Drivrutiner 1(3)

- **En drivrutin är en eller flera funktioner som används för att accessa en extern komponent**
- **En drivrutin gömmer tekniska detaljer**
  - **Användaren ser bara ett API (Application Program Interface), dvs en eller flera C/Assembler funktioner**

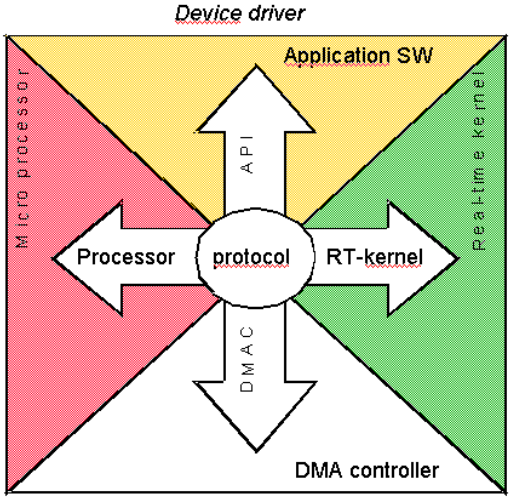
Mittuniversitetet
MID SWEDEN UNIVERSITY

Drivrutiner 2(3)



Mittuniversitetet
MID SWEDEN UNIVERSITY

Drivrutiner 3(3)



Mittuniversitetet
MID SWEDEN UNIVERSITY

Exempel – En AD-omvandlare 1(3)

ADC-12 från Pico Technology

Tekniska data:

Antal kanaler:	1 st
Upplösning:	12 bitar
Insignalområde:	0–5 V
Max sampl.hast:	18 kHz
Repeterbarhet:	±0,5 LSB vid T=25 °C
Överspänningsskydd:	±30 V
Inimpedans:	66 kΩ
Kontaktdon	
A/D-sida:	BNC
PC-sida:	25-pol D-Sub hane

ADC-12 levereras med programvara PicoScope och PicoLog samt drivrutiner för Visual Basic, C++, Delphi, Pascal och Windows (DLL).

Mittuniversitetet
MID SWEDEN UNIVERSITY

Exempel – En AD-omvandlare 2(3)

Utdrag ur den tekniska dokumentationen för ADC-12

The following table specifies the function of each of the routines in the Windows drivers:

Routine	Function
adc10_get_driver_version	Check that this is the correct driver
adc10_open_unit	Open the driver to use a specified printer port
adc10_set_unit	Select which ADC-10 unit to use
adc10_close_unit	Close the specified printer port
adc10_get_value	Get a single reading
adc10_set_trigger	Set a trigger event
adc10_set_interval adc10_get_times_and_values	Set the time interval for the next call to adc10_get_values, or
adc10_get_values	Get a block of readings at fixed intervals
adc10_get_times_and_values	Get a block of readings and their times, at fixed intervals
adc10_get_unit_info	Get information about an ADC10 unit

Mittuniversitetet
MID SWEDEN UNIVERSITY

Exempel – En AD-omvandlare 3(3)

Utdrag ur den tekniska dokumentationen för ADC-12

adc10_get_values

PREF 1 unsigned long PREF2 adc10_get_values (

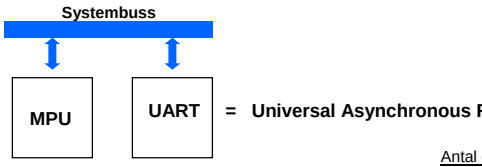
unsigned short HUGE * values,
unsigned long no_of_values);

This routine reads in a block of values. It collects readings at intervals and from channels specified in the most recent adc10_set_interval call.

If a key is pressed while collecting, the routine will return immediately. The return value will be zero if a key was pressed, and the total time in micro-seconds if a block was successfully collected.

Mittuniversitetet
MID SWEDEN UNIVERSITY

Exempel på en enkel drivrutin 1(7)

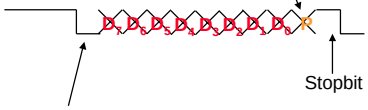


= **Universal Asynchronous Receiver Transmitter**

Antal ettor inkl. paritetsbit: Udda Jämn

Paritetsbit =

1	0	Udda paritet
0	1	Jämn paritet



Hastighet (bps)

1200
2400
4800

Exempel på en enkel drivrutin 2(7)

Programming model

Base + 00	Control
Base + 01	Status
Base + 02	Transmitter data
Base + 03	Receiver data

7	-	
6	-	
5	-	
4	-	
3	-	
2	P	1=>odd , 0=>even
1	sp1	**(sp1 sp0)** (00)=1200, (01)=2400
0	sp0	(10)=4800

7	-	
6	-	
5	-	
4	-	
3	-	
2	ra	1 =>Received byte available
1	te	1 => Transmit register empty
0	pe	1 => Parity error

Exempel på en enkel drivrutin 3(7)

•**Uppgift**

– **Skriv en device driver med följande funktioner (entries) för att stödja seriell kommunikation via UART.**
Metod för parameteröverföringen är valfri.

int initialize(char *base, int speed, int parity);
//base är basadressen till UART. speed är hastighet i
//bps. parity = 0 för jämn och 1 för udda paritet.
//Funktionen returnerar –1 om fel uppstod, annars 0.

int getByte(char *base);
//Nästa mottagna byte (0-255) returneras. Om ej tillgänglig byte så
//returneras –1, om parity error -2.

int putByte(char *base, char item);
//item sänds seriellt. Om det gick bra returneras 0 i annat fall –1.

Exempel på en enkel drivrutin 4(7)

```
initialize:             *A0 contains the base address, D0.W contains the speed in bps
                        * D1.B contains parity, 0=even, 1=odd, D2.W the return value
        MOVE.B    D1,D2
        LSL.B     #2,D2          *Position parity selection bit
        CMP.W     #1200,D0       *Speed = 1200 bps ?
        BNE        next1
        BRA        ok            * 1200, yes
next1:CMP.W       #2400,D0       *Speed = 2400 bps ?
        BNE        next2
        ORI.B     #1,D2
        BRA        ok             * 2400, yes
next2:CMP.W       #4800,D0       *Speed = 4800 bps ?
        BNE        notOk
        ORI.B     #2,D2
        BRA        ok             * 4800, yes
notOk:MOVE.W      #$FFFF,D2
        RTS                      *D2 returns –1 meaning error
ok:   MOVE.B      D2,(A0)        *Control byte to base address + 0
        CLR.W     D2
        RTS                      *D2 returns zero meaning OK
```

Exempel på en enkel drivrutin 5(7)

```
getByte:    *A0 contains the base address, D2.W the return value

        MOVE.W   #$FFFF,D2      *Return code if no byte is available
        BTST.B    #2, 1(A0)     *Byte available?
        BEQ       exit          *Jump if byte is not available
        MOVE.W   #$FFFE,D2      *Return code if parity error
        BTST.B    #0,1(A0)      *Parity error ?
        BNE       exit          *Jump if parity error
        CLR.W     D2
        MOVE.B    3(A0),D2      *Received byte is the return value
exit:   RTS
```

Mittuniversitetet
MID SWEDEN UNIVERSITY

Exempel på en enkel drivrutin 6(7)

```
putByte:    *A0 contains the base address, D0.B the item to be sent
            *D2.W the return value

        MOVE.W   #$FFFF,D2      *Return value = -1 if not ready
        BTST.B    #1,1(A0)      *Is the transmitter ready for a new item?
        BEQ       exit          *No it isn't
        MOVE.B    D0,2(A0)      *Write item to transmitter register
exit:   RTS
```

Mittuniversitetet
MID SWEDEN UNIVERSITY

Exempel på en enkel drivrutin 7(7)

Vi har en serie av tre drivrutiner som döljer detaljer kring UART-hårdvaran för applikationsprogrammeraren.

```
int initialize( char *base, int speed, int parity);
int getByte(char *base);
int putByte(char *base, char item);
```

Vi har skapat ett interface, API mot applikationsprogramvaran

Mittuniversitetet
MID SWEDEN UNIVERSITY