Synkronisering

Ur innehållet:

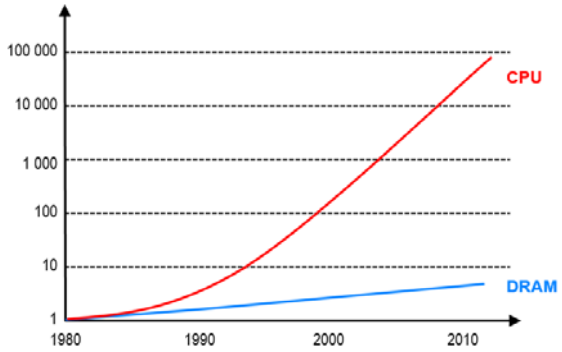
- Arbetstakter - jämförelser
- Repetition: Synkrona/asynkrona gränssnitt
- Tidsdiagram
- "Verklig tid" - räknarkrets "SysTick"
- Tillämpning: LCD-ASCII displaymodul

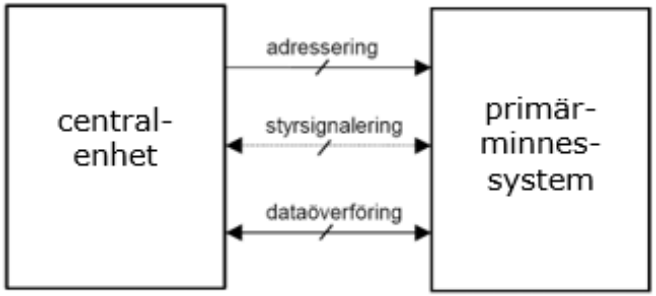
Läsanvisningar:

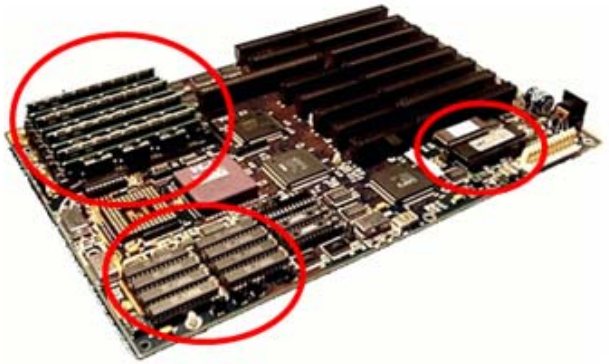
- Arbetsbok avsnitt 5
- Datablad "ASCII-display"

1

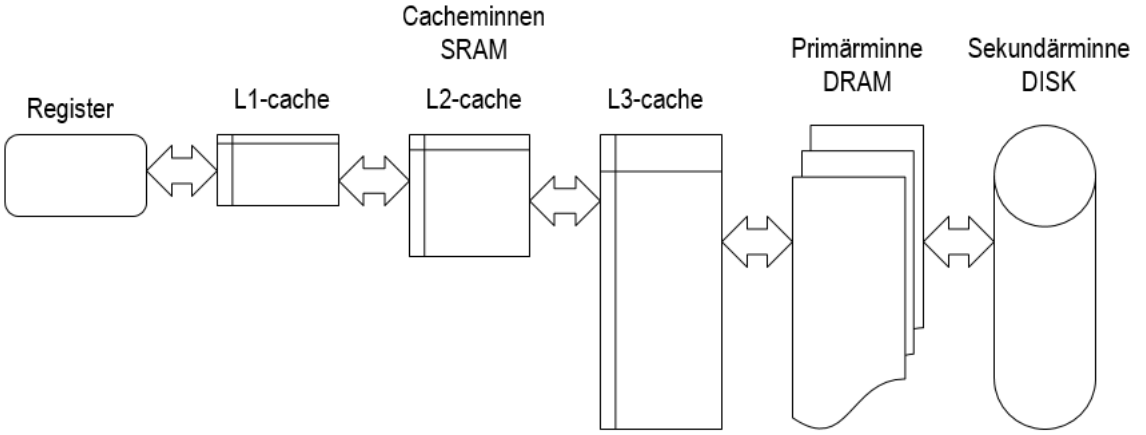
Arbetstakt - typiska åtkomsttider







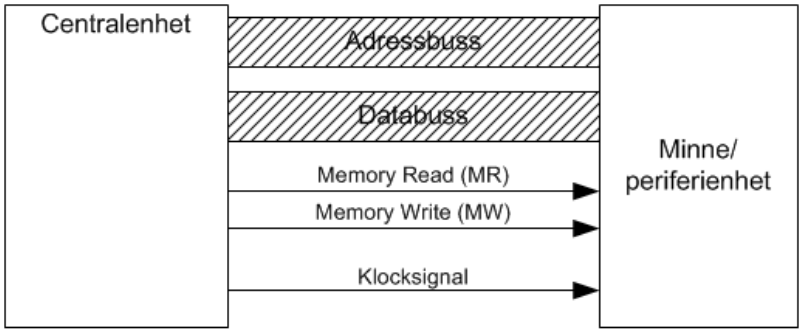
Storlek	1 KB	64 KB	256 KB	2-4 MB	4-16GB	4-16 TB
Åtkomsttid	300 ps	1 ns	3-10 ns	10-20 ns	50-100 ns	5-10 ms



Kraftigt varierande prestanda resulterar i stora skillnader i åtkomsttid

Ovillkorlig överföring

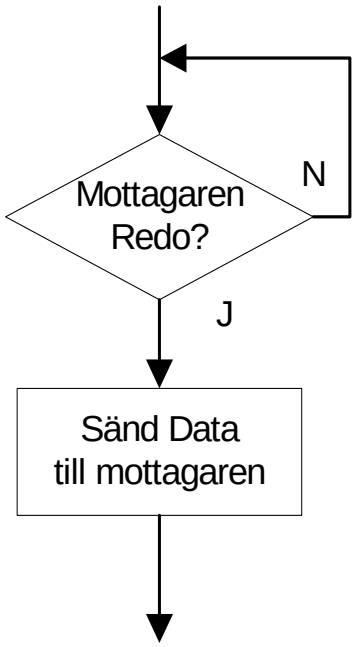
Kräver samtidighet "synkron" - en speciell signal, "Enable", används då för att ange exakt när datautbyte ska ske.



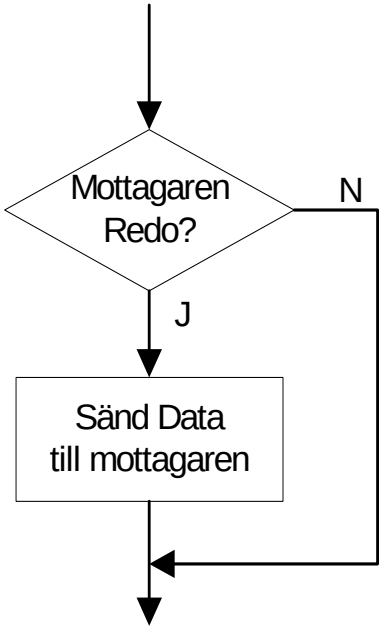
Flankerna definierar samtidigheten. Signalen kallas ofta "klocksignal".

Villkorlig överföring

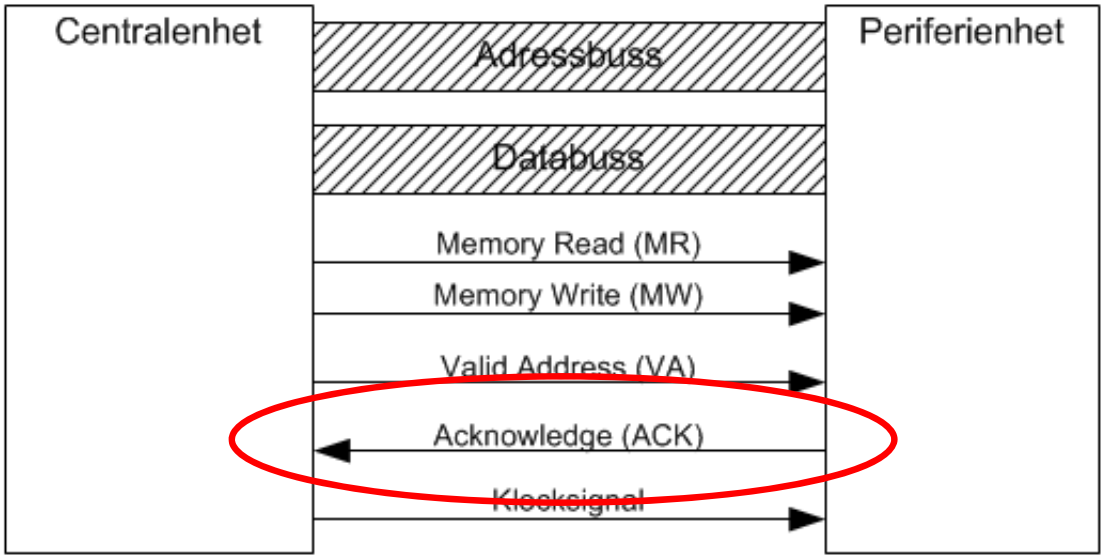
Periferienheten har ett register med en statusbit som anger om data finns eller ej.



Busy Wait

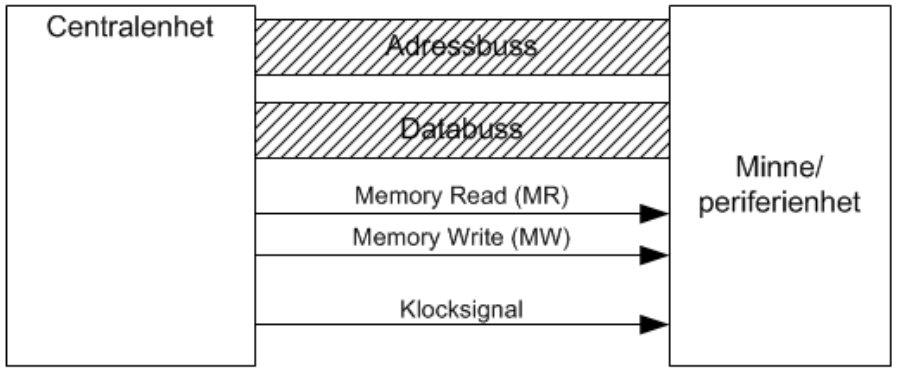


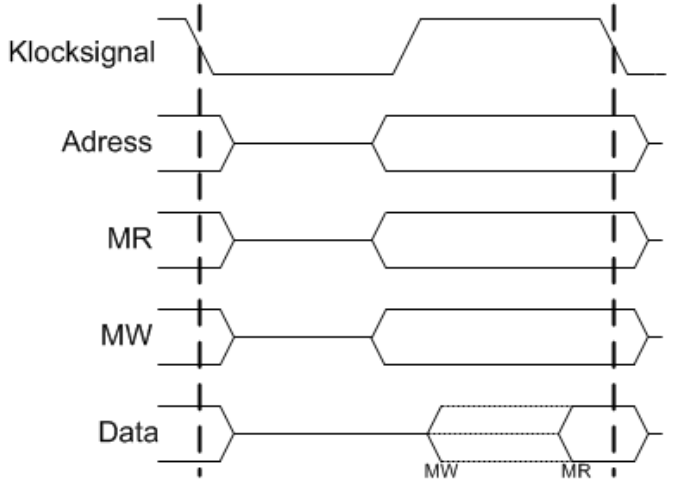
Polling



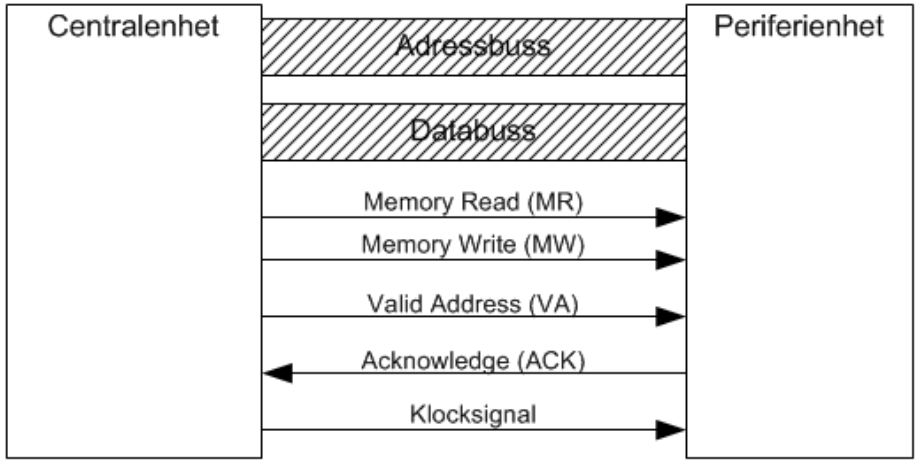
Förutsätter inte samtidighet men kräver speciella "handskakningssignaler".
Ett sådant gränssnitt kallar vi därför "asynkront".

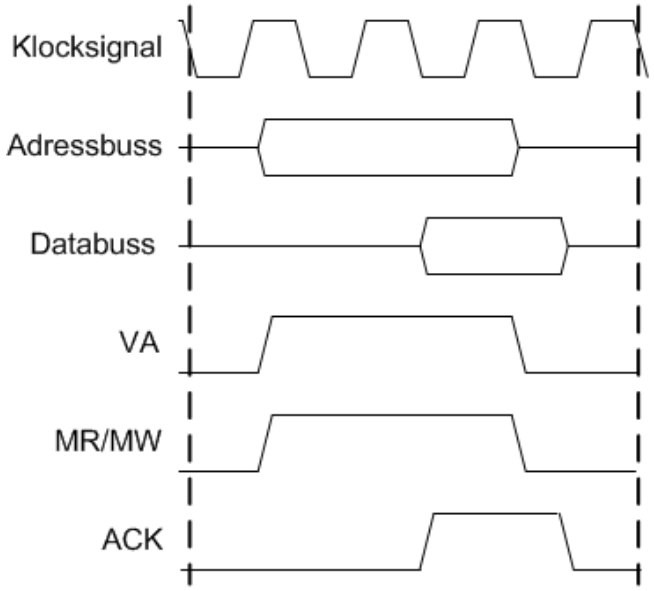
Synkrona och asynkrona gränssnitt





Synkront -
En klocksignal från centralenheten bestämmer när datautbyte sker

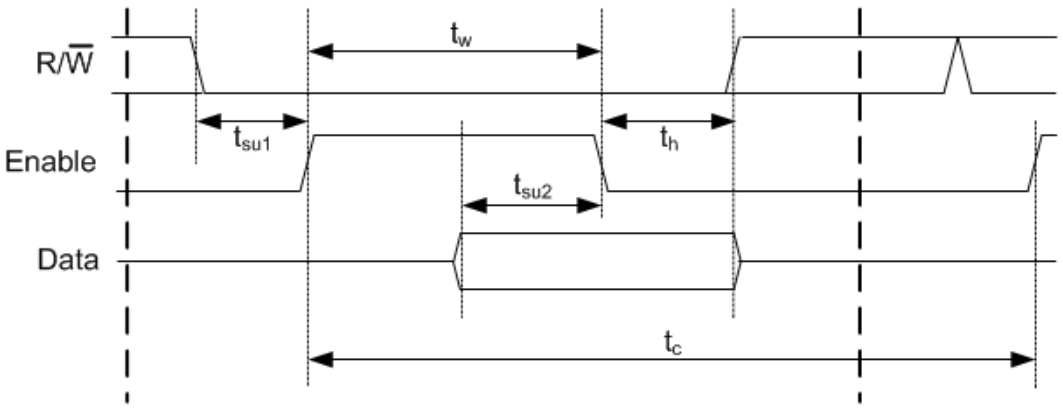




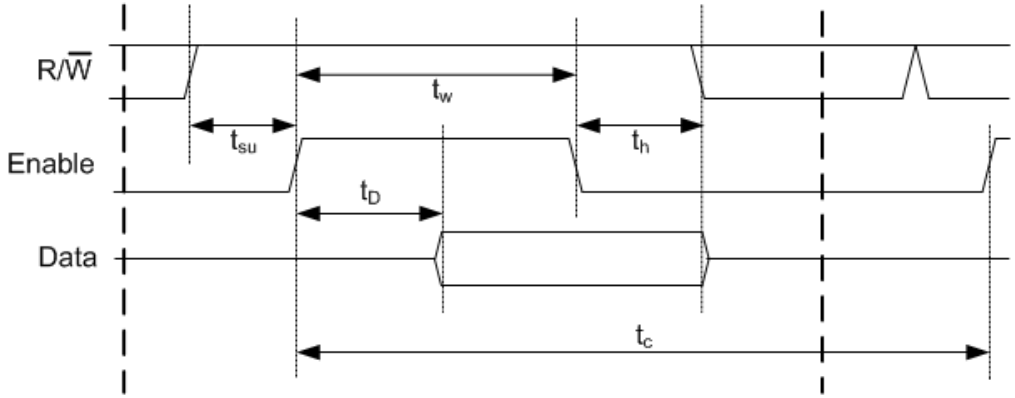
Asynkront -
Handskaknings-signaler bestämmer när datautbyte kan ske

Tidsdiagram - underlag för synkronisering

Skrivcykel - data överförs från CPU till periferienhet



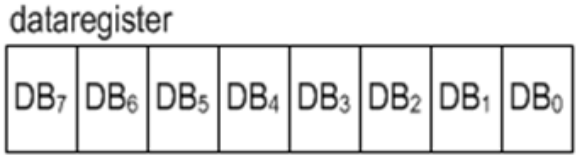
Läscykel - data överförs från periferienhet till CPU

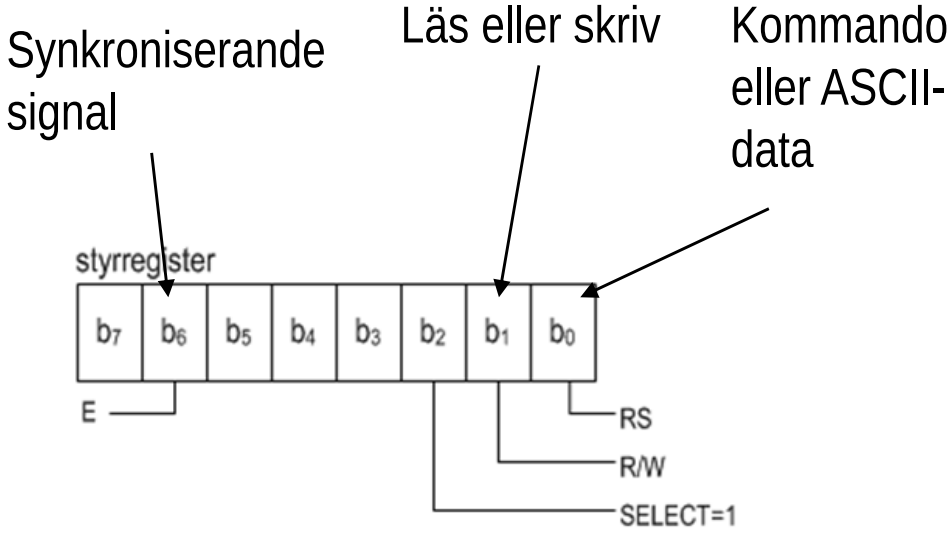


		min
t_c	cykel tid	min
t_w	klockpuls ("Enable") varaktighet (hög och låg)	min
t_{su1}	styrsignalernas setup-tid, före positiv E-flank	min
t_{su2}	setup-tid för data, skrivning, före negativ E-flank	min
t_D	setup-tid för data, läsning, före negativ E-flank	max
t_h	hold-tid, varaktighet (efter negativ E-flank)	min

EXEMPEL - Gränssnitt/algoritm

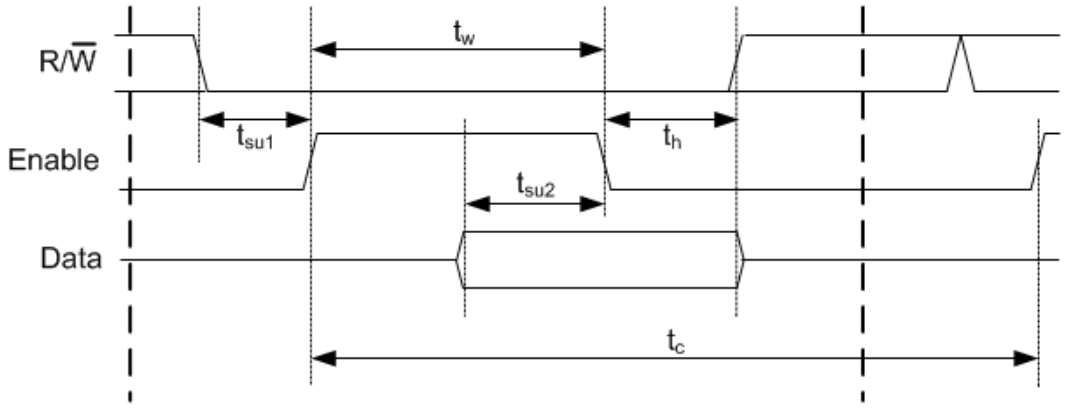
Byte som ska överföras



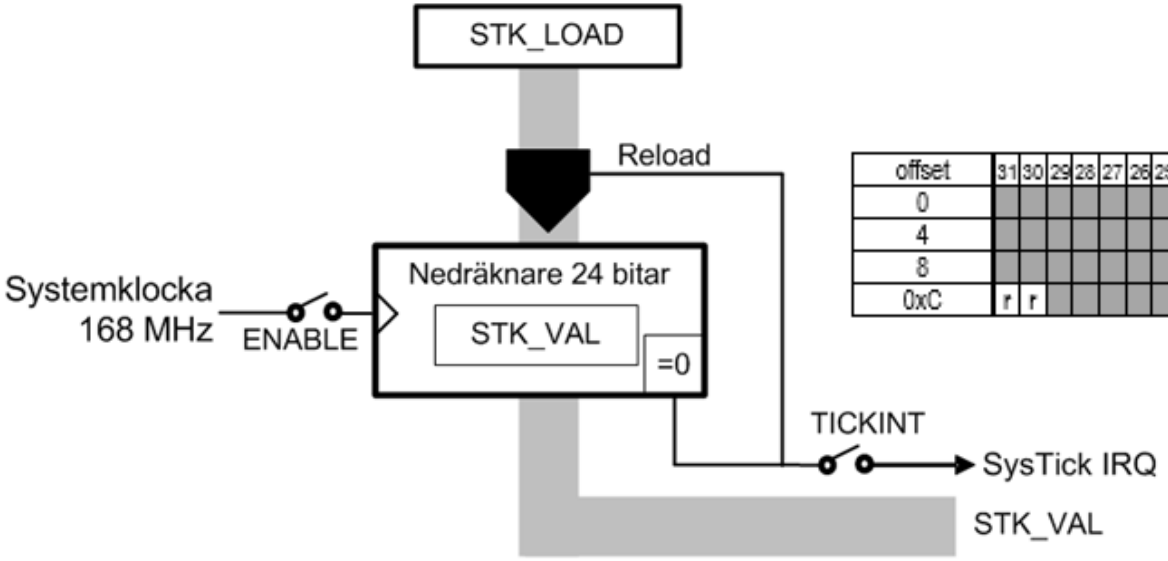


Skriv 8 bitar till controller:

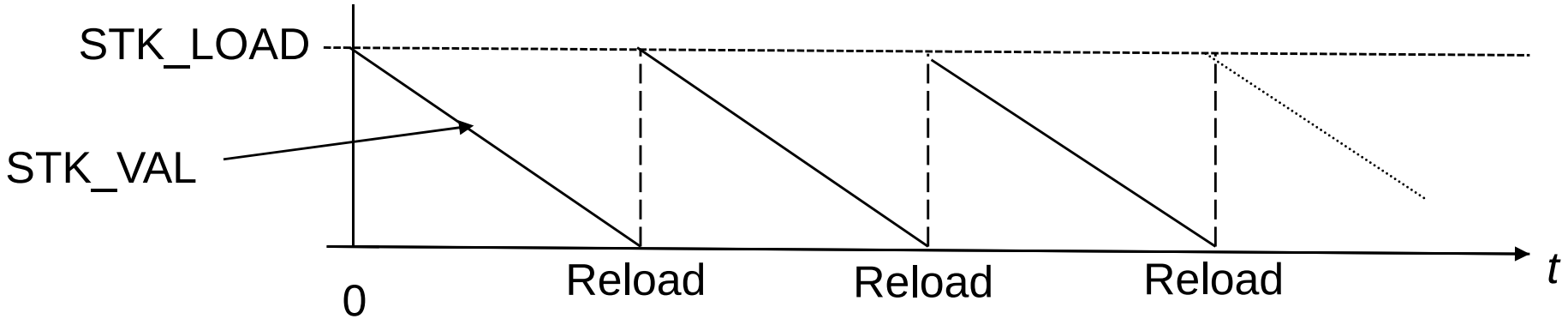
styrregister(RW)=0;
Vänta t_{su1};
styrregister(E)=1;
dataregister = (8 bitar);
Vänta t_{su2}; (dock tills minst t_w förflutit)
E = 0;
vänta tills totalt t_c förflutit;



Räknarkrets - "SysTick"



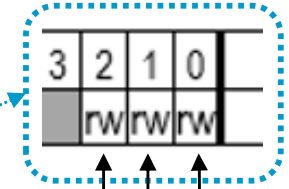
offset	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Register
0																rw															rw	rw	rw	STK_CTRL
4									rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	STK_LOAD
8									rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	STK_VAL
0xC	r	r							r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	STK_CALIB



Räknarkrets - register

STK_CTRL (0xE000E010) Status och styrregister

offset	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Register
0																rw														rw	rw	rw	STK_CTRL



Bit 16: (COUNTFLAG):
Biten är 1 om räknaren räknat ned till 0.
Biten nollställs då registret läses.

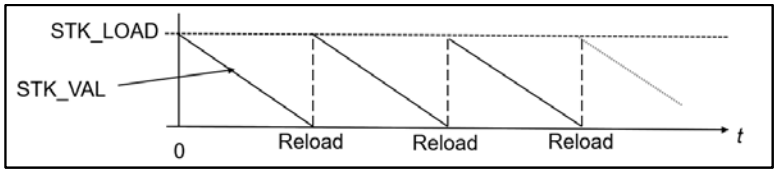
Bit 0: (ENABLE) Aktivera räknare
0: Räknare deaktiverad
1: Räknare aktiverad

Bit 1: (TICKINT): Aktivera avbrott
0: Inget avbrott genereras.
1: Då räknaren slagit om till 0 genererar SysTick avbrott.

Bit 2: (CLKSOURCE) biten är 1 efter RESET:
0: Systemklocka/8 (168Mhz/8 = 21MHz)
1: Systemklocka (168MHz)

STK_LOAD (0xE000E014) Räknarintervall

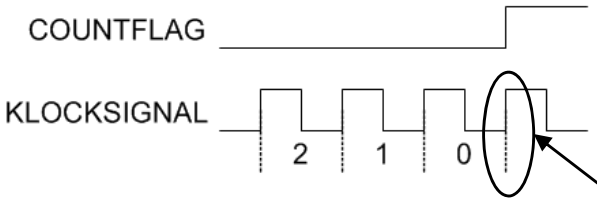
offset	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Register
4									rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	STK_LOAD



Exempel: med 168MHz

Periodtid= $\frac{1}{168\times10^{-6}}$ s ≈ 6ns

För $\frac{168}{168\times10^{-6}}$ får vi räknarintervallet 1µs



Omslaget till 0 sker i slutet av perioden vilket gör att den korrekta initieringen för 1µs fördröjning blir 168-1 klockcykler

Fördröjning med "SysTick"

Skapa en funktion `delay_250ns(void)` som blockerar (fördröjer) den anropande funktionen med minst 250 ns. Visa också hur denna kan användas för att skapa en fördröjningsrutin `delay_mikro( unsigned int us)` som fördröjer programexekveringen variabelt antal mikrosekunder.

Vi löser på tavlan...

Algoritm:

```
STK_CTRL = 0        Återställ SysTick
STK_LOAD = CountValue
STK_VAL = 0         Nollställ räknarregistret
STK_CTRL = 5        Starta om räknaren
Vänta till COUNTFLAG=1
STK_CTRL = 0        Återställ SysTick
```

Periodiskt blinkande diodramp

Vi har en diodramp ansluten till port D (0-7). Demonstrera en applikation som får hela rampen att blinka 1 gång/sekund.
Initieringsrutinen `init_app()`, som initierar porten är given liksom följande portdefinition:

```
#define GPIO_ODR_LOW ((volatile unsigned char *)(0x40020C14))
```

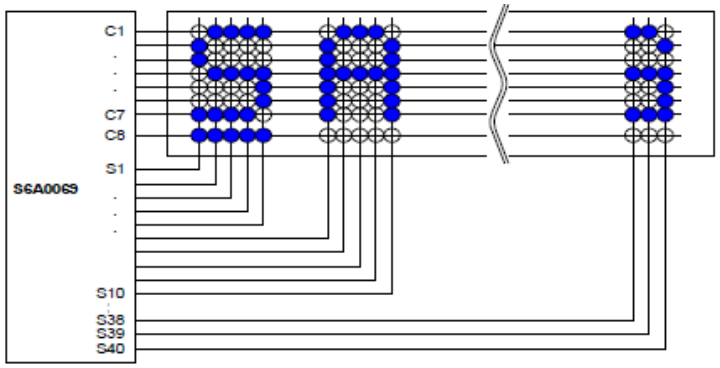
Vi demonstrerar funktionen med simulatorn...

Program har helt olika tidsegenskaper då dom utförs i simulator respektive hårdvara. Använd villkorlig kompilering för att anpassa fördröjningen i exemplet för simulator

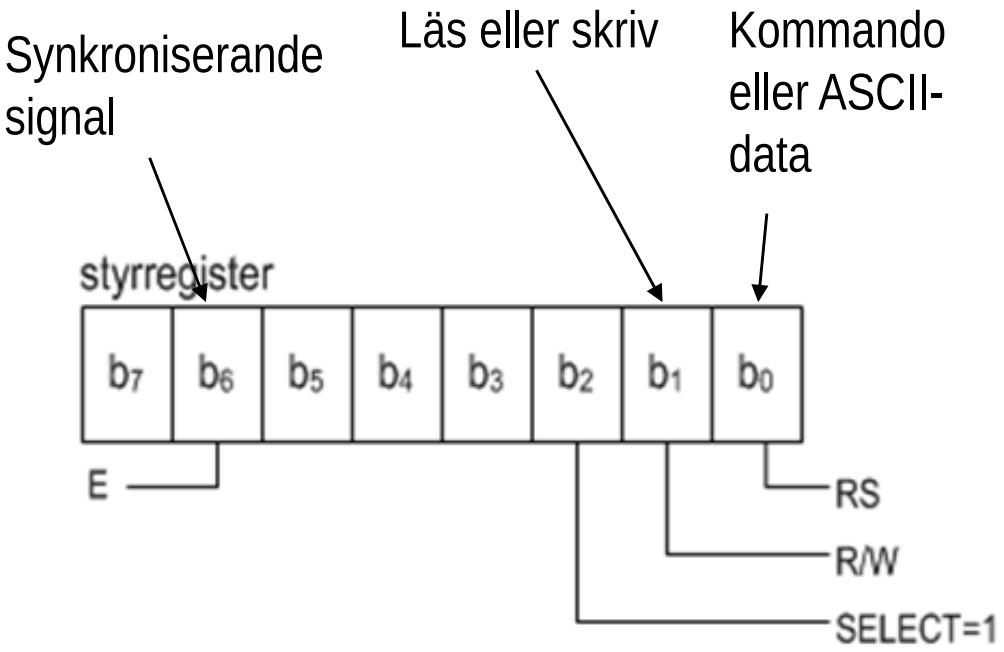
Programmering av ASCII-display

Kretsen översätter ASCII-tecken till motsvarande bit-mönster via ett enkelt gränssnitt:

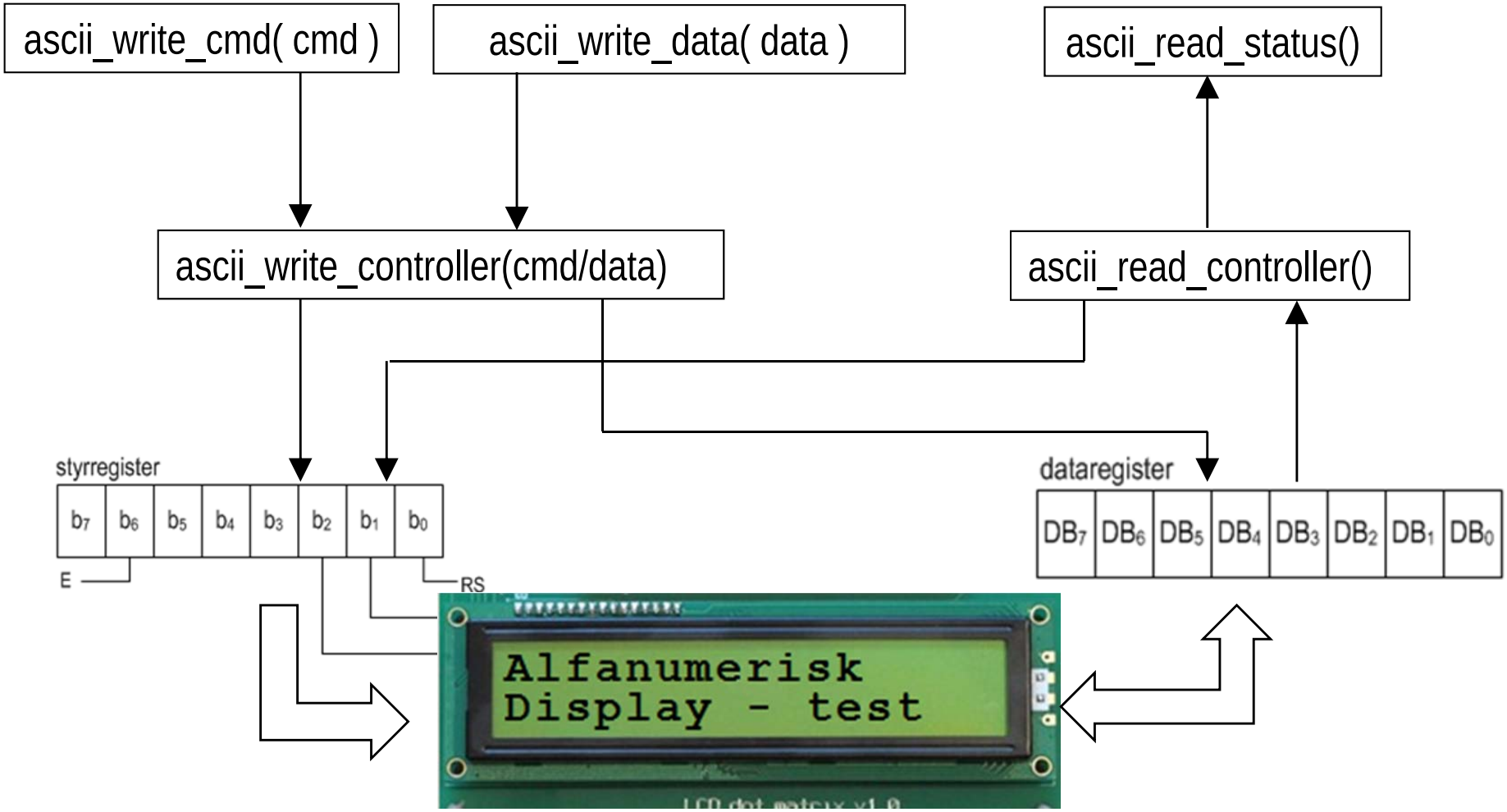
- styrregister - 4 bitar används
- dataregister - 8 bitar används



Byte som ska överföras

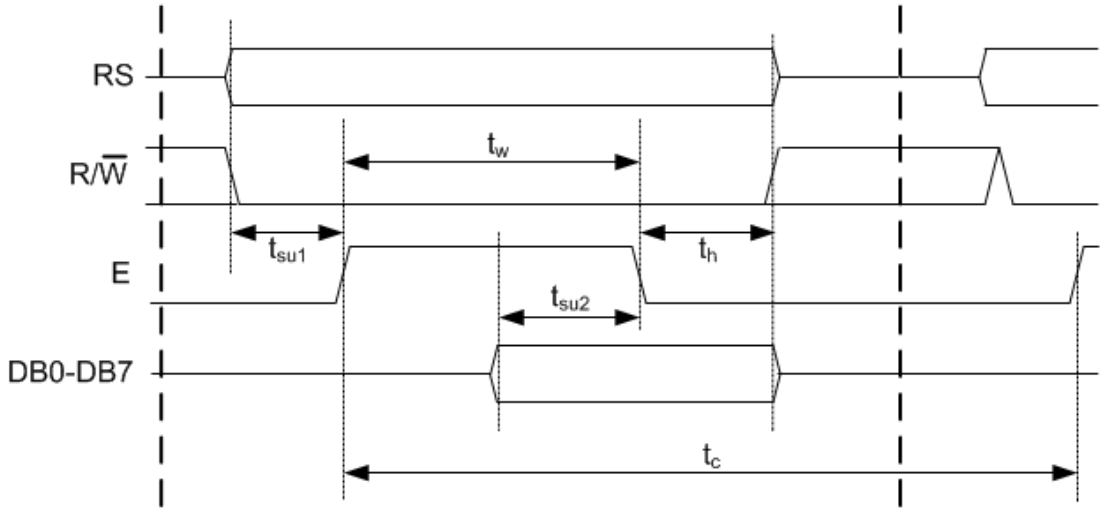


Programstruktur



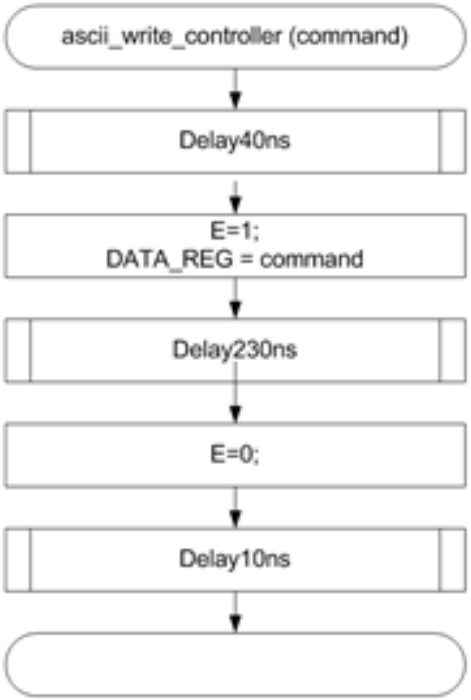
Skrivcykel - karakteristika

Databladets tidsdiagram illustrerar hur styrsignaler och data ska läggas ut



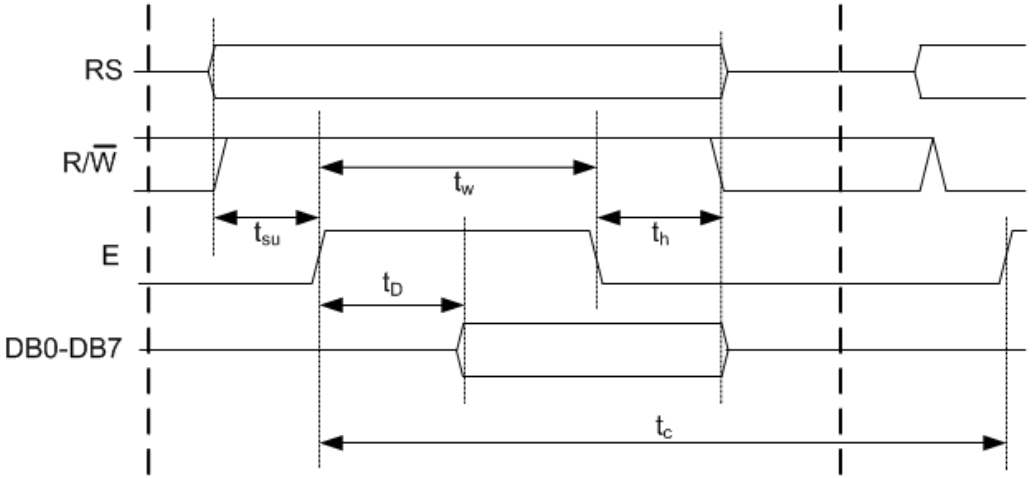
		min
t_c	cykel tid	500 ns
t_w	klockpuls ("Enable") varaktighet (hög och låg)	230 ns
t_{su1}	styrsignalernas setup-tid, före positiv E-flank	40 ns
t_{su2}	setup-tid för data, skrivning, före negativ E-flank	80 ns
t_h	hold-tid, varaktighet (efter negativ E-flank)	10 ns

Algoritm:



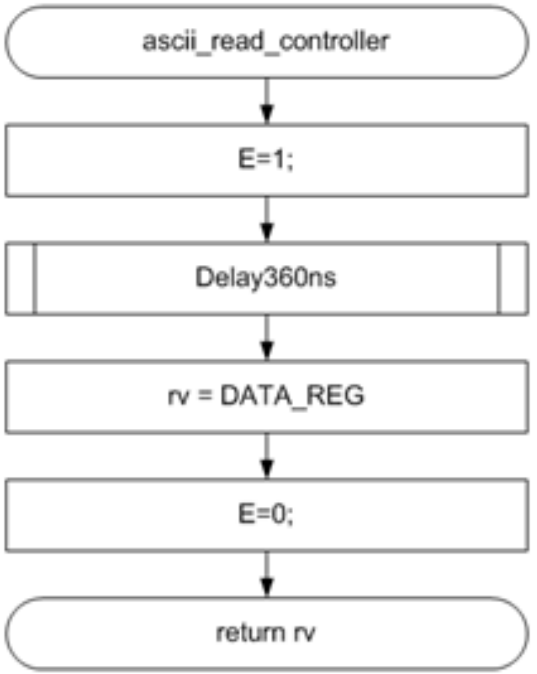
Läscykel - karakteristika

Databladets tidsdiagram illustrerar hur styrsignaler ska läggas ut och hur displaymodulen svarar med data.



		min	max
t_c	cykel tid	1000 ns	
t_w	klockpuls ("Enable") varaktighet (hög och låg)	450 ns	
t_{su1}	styrsignalernas setup-tid, före positiv E-flank	60 ns	
t_D	setup-tid för data, läsning, före negativ E-flank		360 ns
t_h	hold-tid, varaktighet (efter negativ E-flank)	10 ns	

Algoritm:



Programmering av ASCII-Display

Implementera funktioner:

- ascii_write_controller(cmd/data)
- ascii_read_controller()

Vi löser på tavlan

resten av programpaketet lämnas som självverksamhet som förberedelser för laborationen...

Programstruktur