15.1 a)

```
inport  EQU  $0FFF
utport  EQU  $0FFE

prg     LDAA inport
        CMPA #9
        BLS  ok
        LDAA #$0F       *Ingen siffra 0-9
ok      STAA utport
```

b) Detta är ovillkorlig in- och utmatning.

15.2

```
inport  EQU  $0401
utport  EQU  $0400

L0      LDAA #%11000011     * NS:röd-gul, ÖV:gul-grön
        STAA utport
        BSR  DELAY2
        LDAA #%00100100     * NS:grön, ÖV:röd
        STAA utport
        BSR  DELAY40
L1      LDAB inport         * Väntande bilar?
        BITB #%00000011
        BEQ  L1             * Inga bilar väntar
        LDAA #%01100110     * NS:gul-grön, ÖV:röd-gul
        STAA utport
        BSR  DELAY2
        LDAA #%10000001     * NS: röd, ÖV: grön
        STAA utport
        BSR  DELAY40
        BRA  LO
```

15.3

```
inport  EQU  $0400
instat  EQU  $0401
utport  EQU  $0402
utstat  EQU  $0403
antal1  EQU  35
antal2  EQU  $111
datbuf1 EQU  $2400
datbuf2 EQU  $2500
rdymask EQU  %00000001
```

a)

```
prg1    LDAB #antal1
        LDX  #datbuf1
wait1   LDAA instat
        BITA #rdymask
        BEQ  wait1
        LDAA inport
        STAA ,X+
#datbuf2+antal2
        DECB
        BNE  wait1
        ...
```

b)

```
prg2    LDX  #datbuf2
wait2   LDAA utstat
        BITA #rdymask
        BEQ  wait2
        LDAA ,X+
        STAA utport
        CMPX
        BNE  wait2
        ...
```

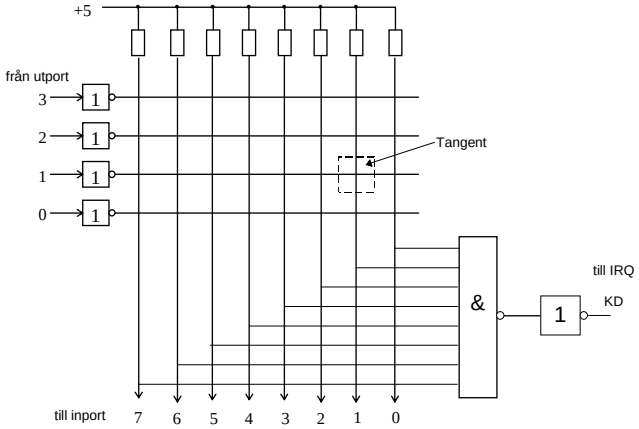
15.4 a)

```
*****************************************************************
*SUBRUTIN - GETCOM
*Beskrivning: Rutinen identifierar och registrerar första nedtryckta tangent.

*Anrop:      BSR GETCOM
*Indata:     Inga
*Utdata:     Tangentnummer = hexadecimalt talpar "radnr:kolumnnr"
*            för nedtryckt tangent i register A.
*Reg-påverkan:Register A
*Anr subr:   DECODE
*****************************************************************
GETCOM  PSHB
        PSHC
*
        LDAB #1
ACTROW  STAB ROWADR     *aktivera raden
        LDAA COLADR     *läs kolumner
        CMPA #$FF       *är någon tangent är nedtryckt?
        BEQ  NXTROW     *om nej, undersök nästa rad
        BSR  DECODE     *om ja, avkoda nedtryckt tangent
        PULC
        PULB
        RTS
NXTROW  LSLB            *nej, byt till nästa rad
        BITB #%0001000  *alla rader har aktiverats?
        BEQ  ACTROW     *nej, aktivera nästa rad
        LDAA #$FF       *ja, returnera FF i A
        RTS
*****************************************************************
 b)
*****************************************************************
*SUBRUTIN - DECODE
*Beskrivning: Rutinen avkodar kolumn- och radmönster, innehållande information
*            om nedtryckt tangent, och identifierar dess läge, som ett hexa-
*            decimalt talpar "radnr:kolumnnr".

*Anrop:      BSR  DECODE
*Indata:     Kolumnmönster i register A och radmönster i register B.
*Utdata:     Tangentläget ”radnr:kolumnnr” i register A.
*Reg-påverkan:CC-, A- och B-registren.
*Anr subr:   Ingen.
*****************************************************************
DECODE  LEAS -2,S       *Skapa plats för 2 bytevariabler på stack
*
        CLR  ,S
        DEC  ,S         *initiering av radnummer-räknare till -1
*
        CLR  1,S
        DEC  1,S        *initiering av kolumnnummer-räknare till -1
*
                        *identifiering av aktiv kolumn
KLOOP   INC  1,S        *öka kolumn-räknare med 1
        RORA            *rotera tills 0 i carry-flagg
        BCS  KLOOP
```

```
*
                                *identifiering av aktiv rad
RLOOP   INC     ,S              *öka rad-räknare med 1
        RORB                    *rotera tills 1 i carry-flag
        BCC     RLOOP
*
        LDAA    ,S+             *radnummer tas omhand
        LSLA                    *skjut radnr 4 steg åt vänster
        LSLA
        LSLA
        LSLA
*
        ADDA    ,S+             *placera in kolumnnr
        RTS
*************************************************************************
```

15.5 Lösningsförslag:

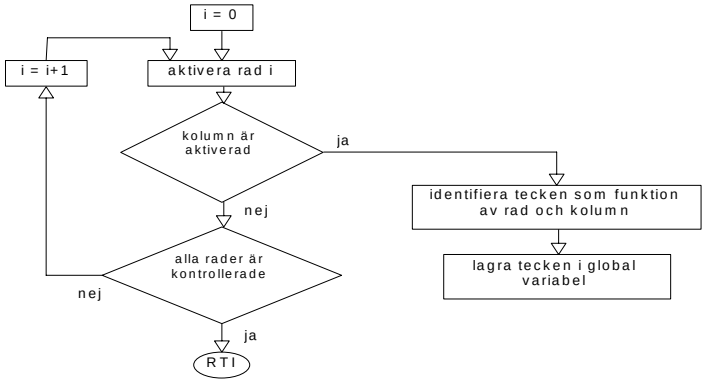
a) Tangentbordet ansluts lämpligen till en utport och en inport, samt



till avbrottsingång på 6809 på följande sätt:

b) 1. tangentnedtryckning upptäcks genom att -aktivera alla radledningar och
- vänta på avbrottsbegäran

2. därefter identifieras, i avbrottsrutinen, den nedtryckta tangenten enligt:



15.6 **a)**

```
INISTR  PSHX
        STX     STRPRINT        * Initiera strängpekare
        LDX     #PRIRQ
        STX     $FFF2           * Initiera IRQ-vektorn
        CLR     $0400           * Nollställ avbrottsvippan
        ANDCC   #$EF            * Tillåt avbrott
        PULX
        RTS
```

b)

```
PRIRQ   LDX     STRPNT          * Hämta pekare
        LDAA    ,X+             * Hämta ASCII-tecken
        BEQ     STREND          * Strängen slut
        STAA    $0400           * Skriv ASCII
        BRA     SLUT
STREND  PULA                    * Hämta CC från stacken
        ORA     #%00010000      * Stäng av IRQ
        PSHA                    * Lägg tillbaka CC
SLUT    STX     STRPNT          * Uppdatera pekare
        RTI
```

c) Ett bra sätt är att låta avbrottsrutinen sätta en flagga i minnet då slutttecknet lästs.

15.7 **a)** WAit for Interrupt.
b) WAI STATUS → stack, väntar på avbrott.
c) Se WAI i instruktionslista, rad 2 i tabellen: VfPPP = 5 st klockcykler.
Gäller vid både IRQ och XIRQ

15.8 1) Vid ett omaskerat avbrott kommer startar avbrottsrutinen efter 6 st klockcykler = fVfPPP.
2) Vid ett maskerat avbrott fortsätter processorn med nästa instruktion efter 2 st klockcykler. Detta kan användas för att synkronisera exekveringens fortsättning till en yttre händelse.

15.9 Sändaren har klockfrekvensen f_{TC} = 10 kHz. Mottagaren samplar insignalen med frekvensen f_m. Första samplet tas vid t = ε. Bitidentifieringsögonblicken ges dedan av

t_{id}:, ε+(k/2)·t_{RC}, ε+(3k/2)·t_{RC}, ... (k = 16)

där ε är tiden mellan negativ flank hos insignalen och negativ flank hos f_m. Bitidentifieringstiderna kan skrivas

t_{id} = ε + (1/2 + n)·k·t_{RC}

a) Samplingen på mottagarsidan kommer att göras med för korta intervall. Värsta fallet fås därför för ε = 0.
Mottagarens bithastighet: $f_m = 1.07 \cdot f_{TC} = 10.7$ kHz
$f_{RC} = k \cdot f_m = 171.2$ kHz $t_{RC} = 5.84 \cdot 10^{-6} = 5.84$ µs
$t_m = k \cdot t_{RC} = 93.46$ µs

Följande bitidentifieringstidpunkter erhålles (µs).

46.7, 140.2, 233.6, 327.1, 420.6, 514.0, 607.5, 700.3, 794.4, 887.9, 981.3

b) Den sända paritetsbiten tolkas som stoppbit. Om den sända paritetsbiten är en nolla, upptäcks felet som "framing error". Vi ser att den sända bit 7 läses två gånger. Andra gången förväntar sig mottagaren en paritetsbit. Paritetsfel signaleras om sänd bit 7 och sänd paritetsbit har olika värden. Om båda är 1, så erhålles varken ”framing error” eller paritetsfel. Då tolkas bitvärdena korrekt.

c) Nu är värsta fallet att mottagarens räknare börjar räkna en period (av f_{RC}) efter startbitens negativa flank (ε = t_{RC}).

Mottagarens bithastighet: $f_m = 0.93 \cdot f_{TC} = 9.3$ kHz
$f_{RC} = k \cdot f_m = 148.8$ kHz $t_{RC} = 6.72 \cdot 10^{-6} = 6.72$ µs
$t_m = k \cdot t_{RC} = 107.5$ µs

Detta ger samplingstiderna

60.5, 168.0, 275.5, 383.0, 490.6, 598.1, 705.6, 813.1, 920.6, 1028, 1136

d) Redan bit 6 kommer att läsas för sent. Mottagaren kommer emellertid att få en korrekt stoppbit om inte ett nytt ord sänds direkt efter det aktuella. Då läses startbiten som stoppbit => felet upptäcks ("framing error"). Beroende på hur bitarna i det mottagna ordet ser ut, kan paritetsfel uppstå.

15.10 a)

```
in      LDAA    indata          * Läs från serieinterfacet
        LDX     inbufpek
        STAA    ,X+             * Skriv till buffert
        STX     inbufpek        * Öka inbufpek med 1
        RTI
```

b)

```
init    PSHX
        PSHC
        LDX     #in             * Adressen till avbr.rut.
        STX     $FFF2
        LDX     #inbp           * Början på inbufpek
        STX     inbufpek        * Begynnelseadr. för pekare
        PULC
        PULX
        ANDCC   #%11101111      * Tillåt IRQ
        RTS
```

15.11 a)

```
ut      LDX     utbufpek
        LDAA    ,X+
        STAA    utdata
        STX     utbufpek
        RTI
```

b)

```
init    PSHX
        PSHC
        LDX     #UT             * Adressen till avbr.rut.
        STX     $FFF2
        LDX     #utbp           * Början på utbufpek
        STX     utbufpek        * Begynnelseadr. för pekare
        PULC
        PULX
        ANDCC   #%11101111      * Tillåt IRQ
        RTS
```

15.12 a)

```
mask    EQU     %00000001

irqhnd  LDAA    instatus        * Inmatningsavbrott?
        BITA    #mask
        BNE     in
        LDAA    utsatus         * Utmatningsavbrott?
        BITA    #mask
        BEQ     ut              * 1 => ej IRQ från utport
        JSR     fel             * Falskt avbrott!
        BRA     slut
in      JSR     inmatn
        BRA     slut
ut      JSR     utmatn
slut    RTI
```

b)

```
in      PSHA
        PSHX
        PSHC                    * Spara register som ändras
        LDAA    indata          * Hämta tecken, nollställ inflagga
        LDX     inbufpek        * Hämta buffertpekare
        STAA    ,X+             * Lagra tecknet, öka pekare
        STX     inbufpek        * Uppdatera pekarvariabeln
        PULC
        PULX
        PULA
        RTS
```

```
ut      PSHA
        PSHX
        PSHC                    * Spara register som ändras
        LDX     utbufpek        * Hämta buffertpekare
        LDAA    ,X+             * Hämta tecken, öka pekaren
        STAA    utdata          * Tecken till utdataregistret
        STX     utbufpek        * Uppdatera buffertpekaren
        PULC                    * Återställ använda register
        PULX
        PULA
        RTS
```

c)

```
init    PSHX
        PSHC                    * Spara register som ändras
        LDX     #inbp           * Ställ in början på inbuffert
        STX     inbufpek
        LDX     #utbp           * Ställ in början på utbuffert
        STX     utbufpek
        LDX     #irqhnd         * Adressen till avbr.rut.
        STX     $FFF2           * Lagra adressen
        PULC                    * Återställ register
        PULX
        ANDCC   #%11101111      * Tillåt IRQ
        RTS
```

15.13 Motparten* har program som tar emot data. Under mottagningen upptäcker motparten att den ej kan ta hand om alla ord som sänds från 6809-systemet. Motparten sänder i god tid XOFF till 6809-systemet. När motparten kan ta emot data igen, sänder den XON. Vi antar att all sändning och mottagning i 6809-systemet är avbrottsstyrd.

*Kan ex vis vara en skrivare där tecken placeras i en buffert som håller på att fyllas.

```
XON     EQU  $11
XOFF    EQU  $13

test_brk PSHA
         PSHC
         LDAA   indata          * Läs ett ord från ACIA:n
         CMPA   #XOFF           * Nej, sändning kan göras
         BNE    exit
wait     LDAA   status          * Mottagaren upptagen
         BITA   #%00000001      * Vänta på data fr. mottagaren
         BEQ    wait
         LDAA   indata
         CMPA   #XON            * Var det XON?
         BNE    wait            * Nej, fortsätt vänta
exit     PULC                   * OK att sända.
         PULA
         RTS
```

15.14

```
TDR             EQU     $0E81
RDR             EQU     $0E81
CR              EQU     $0E80
SR              EQU     $0E80

IBUFPTR         RMB     2               * Plats för pekare till inbuffert
```

Initieringsrutin för ACIA:n

```
IACIA   LDAA    #%00000011      * Master Reset
        STAA    CR
        LDAA    #%10000001      * Avbrott vid mottagning, ej vid
```

sändning;

```
                                *  * 7 bitar, jämn paritet, 2 stoppbitar;
                                *  * E-klockan delas med 16
        STAA    CR
        RTS
```

Avbrottsrutin som tar emot tecken och lägger i inbuffert.
Ingen felhantering görs.

```
IRQRUT  LDAA    SR
        BITA    #%00000001      * Mottagarregister fullt?
        BEQ     EJMOT
        LDAA    RDR             * Hämta mottaget tecken
        LDX     IBUFPTR
        STAA    ,X+             * Lagra i inbuffert
        STX     IBUFPTR
EJMOT   RTI
```

Allmän initiering (vid reset t ex - ingår ej i uppgiften)

```
INIT    LDX     #$1F00
        STX     IBUFPTR         * Initiera pekare till inbuffert
        JSR     IACIA
        LDX     #IRQRUT
        STX     $FFF2
        ANDCC   #%11101111      * Aktivera IRQ
        RTS
```

15.15 -