

# AN-00-017

## MC68

### Duartprogrammering



MC68 är bestyckad med två seriekommunikationskretsar (DUART) varav den första (DUART PORT A) används av DB68. DUART port B är dock tillgänglig och i denna applikationsnot ska vi demonstrera hur porten kan användas.

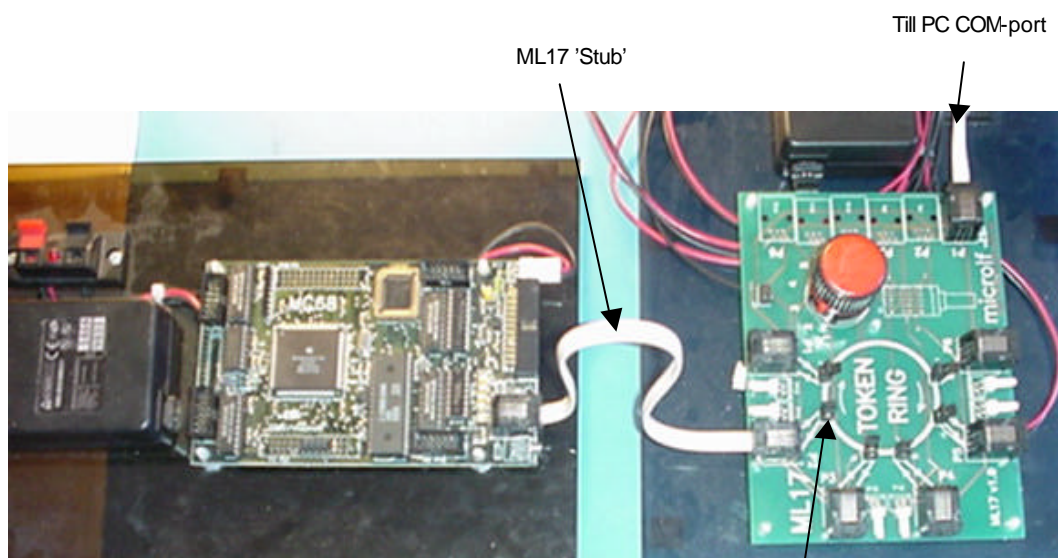
AN-00-017 innehåller:

AN-00-016.PDF (denna beskrivning)  
DUARTPROG.ZIP källtexter

Det första exemplet visar hur enkla drivrutiner kan skrivas helt och hållet i C (simple.m68) nästa exempel är något mer avancerat och illustrerar hur serieporten programmeras för avbrott.

## Förberedelser

MC68 har endast en anslutning (RJ45) för båda serieportarna. För en utförlig beskrivning av konfigurationen hänvisas till MC68 hårdvarubeskrivning. För denna not har vi använt en mycket enkel uppkoppling med MC68 och ML17:



Port B kopplas till ringen, port A till PC-COM-port. Övriga byglar ställs så att ringen sluts.

Den visade uppkopplingen är den enklast tänkbara där vi egentligen bara kopplat ihop Rx och Tx för DUART port B, upp till 6 st MC68'or kan dock anslutas via ML17 och på så sätt fås att realisera en ring-buss.

**simple.m68**

Öppna projektet "build\simple.m68"

Lägg till filen "src\duartc.c" (bara första gången)

```
/*
    DUARTC.C
    MC68 DUART port B
    EJ AVBROTT
*/
```

Portdefinitioner för MC68 finns definierade i "machine.h"

#include &lt;machine.h&gt;

För att göra C-koden nedan mer lättläst definierar vi här makro's för de olika register vi använder...

```
#define DB_CRB      *( (char *) mc68_duart_crb )
#define DB_MR1B     *( (char *) mc68_duart_mr1b )
#define DB_MR2B     *( (char *) mc68_duart_mr2b )
#define DB_CSRB     *( (char *) mc68_duart_csr_b )

#define DB_TXB      *( (char *) mc68_duart_tbb )
#define DB_RXB      *( (char *) mc68_duart_rbb )
```

Vi gör följande standarinitiering av serieporten...

```
void init_duartB( void )
{
    DB_CRB = 0x30;
    DB_MR1B = 0x13;
    DB_MR2B = 0x7;
    DB_CSRB = 0xBB;
    DB_CRB = 0x45;
}
```

Skrivrutin...

```
void write_duartB( int c )
{
    while( (DB_CRB) & 4 );           /* vänta tills TDR ledigt */
    DB_TXB = (char) c;
}
```

Läsrutin...

```
int read_duartB( void )
{
    while( ((DB_CRB)&1 == 0));       /* vänta tills tecken finns */
    return (int) (DB_RXB);
}
```

Huvudprogram för teständamål. Vi använder \_inchar och \_outchar, dvs rutiner för DUART port A, ett tecken läses (från terminalen), skrivs till DUART port B, därefter väntar programmet på att tecknet ska tas emot igen i port B och slutligen skrivs det till terminalen. Testa genom att skriva olika bokstäver på tangentbordet, de ska komma tillbaks fast ett steg försenat...

```
void main()
{
    int c;

    init_duartB();
    while(1){
        c = _inchar();           /* Läs från DUART port A */
        write_duartB( c );       /* skriv till DUART port B */
        c = read_duartB();       /* läs efter ett varv på ringen */
        _outchar( c );           /* skriv till port A */
    }
}
```

**duart-irq.m68**

Öppna projektet "build\duart-irq.m68"

Lägg till filerna "src\dread.c" och "src\lowduart.s68" (bara första gången)

Exemplet är mer komplicerat än det första och vi kommenterar därför bara huvudprogrammet.

While-loopen läser tecken från terminalen och skriver dessa till port B ända tills tecknet '?' skrivs. Tecknen kommer att skickas via ringen och tas om hand av avbrottsrutinen (lowduart.s68) som sparar dessa i en intern buffert.

```
while(1){
    c = _inchar();
    _outchar(c);
    if(c=='?') break;
    i++;
    if(c != '\n')
        write_duart_B(c);
}
```

Då tecknet '?' skrivs kommer slingan att avbrytas varefter följande utskrifter sker:

```
putchar('R');
putchar(recirq+'0');
putchar('\n');
```

dvs antalet 'Rx' avbrott som betjänats (högst 9)

```
putchar('T');
putchar(trirq+'0');
putchar('\n');
```

dvs antalet 'Tx' avbrott som betjänats (högst 9)

slutligen läses avbrottsbufferten till lokala variabeln 'buffer' med read\_duartB. Funktionen returnerar antalet tecken till variabeln j som vi använder för att sätta en slutmarkering innan innehållet skrivs ut till terminalen.

```
j = read_duart_B(i,buffer);
buffer[j]='\0';
puts(buffer);
```

**Läs mer...**

MC68's DUART beskrivs utförligt i läroboken "Datorteknik för högskolans ingenjörsutbildningar" Johansson/Snedsbøl.

Laborationer finner du i "Arbetsbok för laborationer i Datakommunikation"

Följande applikationsnoter behandlar maskinnära programmering i C

AN-00-004	MC68/ML13, program i C och assembler
AN-00-008	MC68 bootloader
AN-00-009	RTK 68 Realtidskärna för MC68
AN-00-014	X68c – enkla exempel
AN-00-015	CAN-kommunikation med MC68/ML12
AN-00-016	Biblioteksrutiner för X68C

Applikationsnoterna finner du på Internetadressen [www.gmv.nu](http://www.gmv.nu).