ML12

CAN Kommunikationskort

Dokument: ML12 - Hårdvarubeskrivning

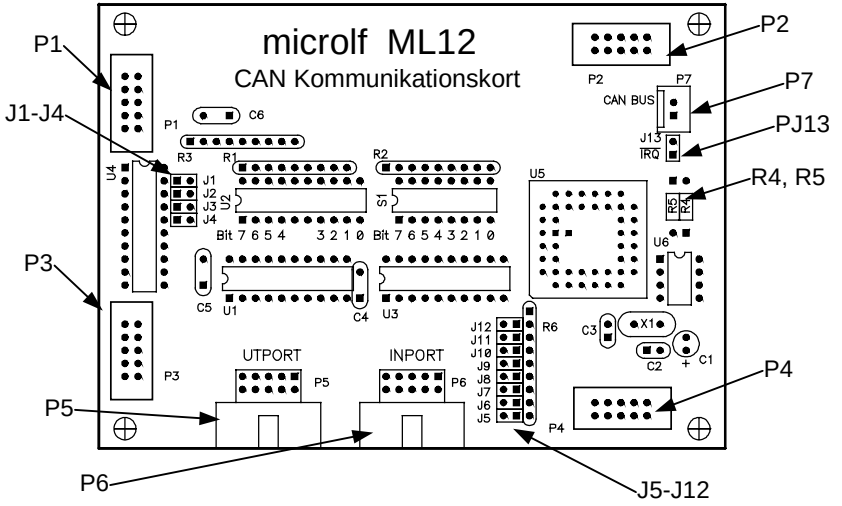
Id. nummer: 131-06

ML12 är ett laborations- och utvecklingskort för CAN-protokollet. Kortet som är anpassat för MC68:s expansionsbuss innehåller även en 8-bitars parallell inport med strömställare och en 8-bitars parallell utport med lysdioder för att kunna ge indata till och studera utdata från kortet.

Den senaste versionen av denna dokumentation finns tillgänglig på Internetadressen:
http://www.gbgmv.se

1. Inledning

ML12 är ett laborationskort för CAN-protokollet och anpassad för MC68's expansionsbuss. Utöver en CAN-krets är kortet bestyckat med en 8-bitars parallell inport och en 8-bitars parallell utport. Indata kan ställas med hjälp av strömställare på inporten och utdata kan avläsas på lysdioder på utporten. Vidare har de parallella portarna anslutningar anpassats för tangentbordet ML23 (Se under Anslutningar nedan). Det finns också en stiftlist som kan byglas för att exempelvis ge kortet olika CAN-adresser eller nod-nummer i distribuerade applikationer.



2. Adressrum

Kortet kan byglas för olika adresser. Se nedan.

2.1. Basadresser

De olika basadresserna ges i tabellen nedan. Studera figur 1 för att lokalisera bygel J1 och J2. (U=Ute, I=Inne)

Tabell 1. Basadresser		
Bygel		Bas adress (-maxadr)
J2	J1	
U	U	$88000 (-$881FF)
U	I	$88200 (-$883FF)
I	U	$88400 (-$885FF)
I	I	$88600 (-$887FF)

Vid leverans är bygel J1 och J2 ute och default är kortets basadress således $88000. Observera att ofullständig adressavkodning används vilket innebär att hela adressområdet från $88000 till $881FF utnyttjas

2.2. I/O-adresser

Följande adresser används på ML12.

Tabell 2. IO-adresser		
Krets	Basadress	Skriv/Läs
CAN-krets	Bas +$000	
Utport	Bas +$100	Skriv
Inport	Bas +$100	Läs

CAN-kretsen upptar hela adressutrummet från $000 $0ff och Ut och inporten nås inom hela adressområde från $100 till $1ff.

3. Kortets funktion

De parallella in- och utgångarna är enkla I/O. Som inport används en buffert 74HC245 och som utport ett register 74HC273. Observera portarnas anslutningar P5 och P6 är anpassade för att anslutas till tangentbordsdelen på ML23.

Kortet är bestyckad med en CAN-krets från INTEL 82527. Denna är byglad för 8-bit Non-Multiplexed Mode och en synkron buss (DSACK används EJ). Det hänvisas här till databladen för 82527 för registerbeskrivning.

Som drivkrets för CAN-bussen används PHILIPS 82C250. Denna har en justerbar SLOPE via spänningsdelaren R4/R5. Vid leverans är denna byglad till jord via R5, som är dragen som en ledningsbana på kortets lödsida. Studera figur 1 och lokalisera R5. För att ändra SLOPE krävs att ledningsbanan på kortets lödsida kapas och önskade motståndsvärden för R4/R5 löd in.

Byglarna J5-J12 är direkt anslutna till CAN-kretsens port 2 och bestyckade med ett pull-up motstånd till +5V. Porten är vid RESET definierad som inport och kan byglas till jord för att ställa indata. Byglarna kan således vara användbara för att ge kortet ett unikt nummer etc. Eller för att starta upp olika programrutiner från PROM exempelvis.

4. Kortets anslutningar

Studera figur 1 som visar kortets olika anslutningar.

4.1. Expansionsbuss

Anslutningarna P1, P2, P3 och P4 (alla 10-poliga) utgör kortets expansionsbuss.

Tabell 3. Expansionsbuss

Pin	P1	P2	P3	P4
1	NC	GND	NC	GND
2	CSEXT	D0	A15	A7
3	R/W	D1	A14	A6
4	AS	D2	A13	A5
5	RESET	D3	A12	A4
6	CLKOUT	D4	A11	A3
7	TIN1	D5	A10	A2
8	TOUT1	D6	A9	A1
9	TGATE1	D7	A8	A0
10	NC	+5V	NC	+5V

4.2. I/O

Anslutning P5 (Parallell Output) och P6 (Parallell Input) beskrivs i tabellen nedan.

Tabell 4.
I/O-anslutning

Pin	P5	P6
1	GND	GND
2	DO0	DI0
3	DO1	DI1
4	DO2	DI2
5	DO3	DI3
6	DO4	DI4
7	DO5	DI5
8	DO6	DI6
9	DO7	DI7
10	+5V	+5V

Anslutning P7 är för CAN-bussen.

Tabell 5.
CAN-anslutning

Pin	P7
1	CAN-Low
2	CAN-High

4.3. Avbrott

Avbrott från CAN-kretsen kan kopplas vidare från anslutning J13.

4.4. Kortets byglar

Tabell 6. Byglar

Bygel	Funktion
J1-J2	Anger kortets basadress, se kapitel 2
J3-J4	För framtida bruk
J5-J12	Port P2.0 -P2.7 på CAN-krets 82527

5. Applikationsexempel

Följande subrutiner ger exempel på hur ML12 kan användas tillsammans med MC68. Programexemplet är utvecklat med ”X68C”.

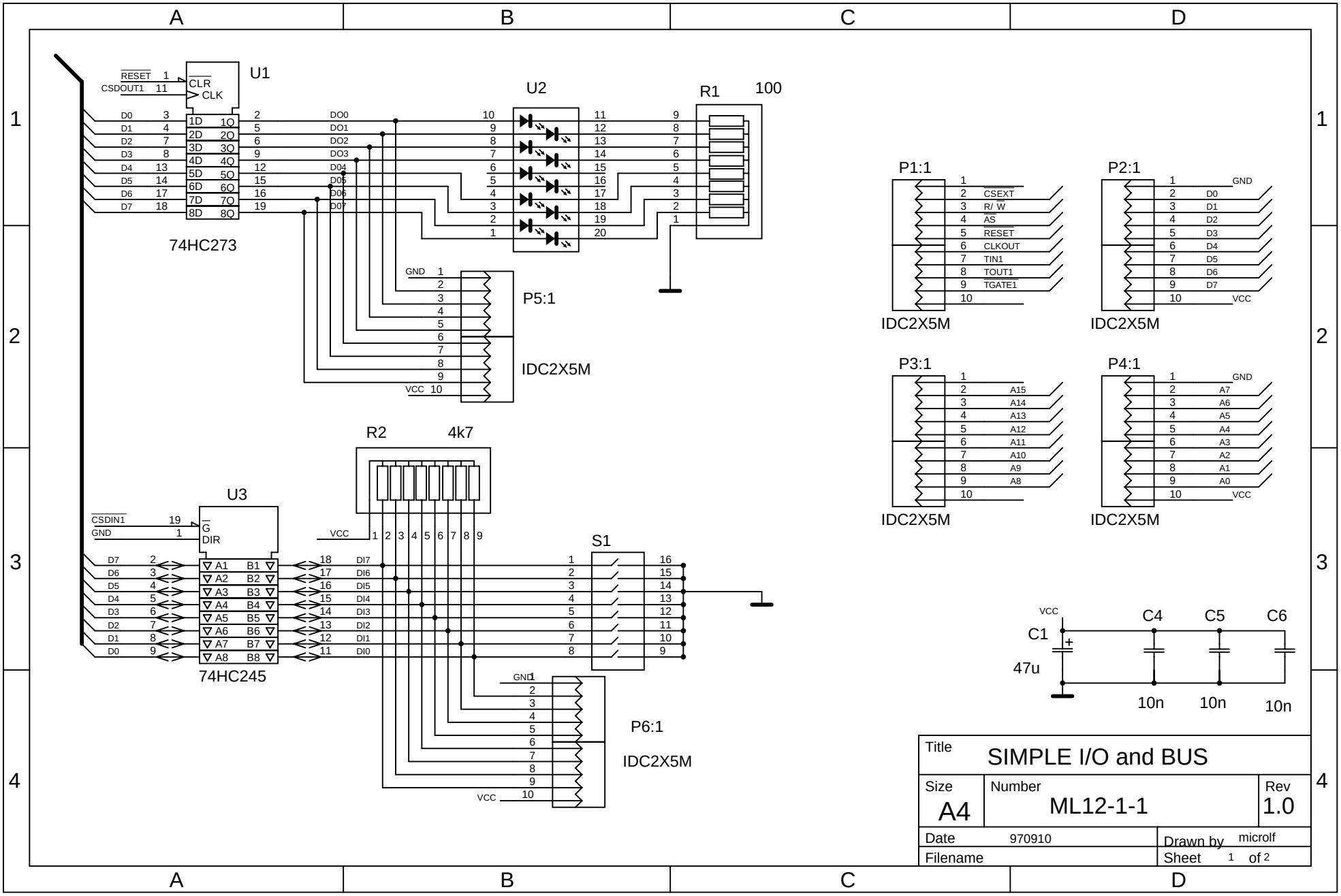
```
*
*      Interface routines MC68/ML9

* C-prototypes:
*      void   CANinit(void);
*      void   CANSend(int msgobject, char *data);
*      void   CANSetupRec(int msgobj);
*      int    CANrec(int msgobject);

*      Export ...
       DEFINE _CANInit
       DEFINE _CANSend
       DEFINE _CANSetupRec
       DEFINE _CANRec


* io-addresses for MC68/ML12
CAN    EQU    $88000

**************************************
* CAN controller init
* see Intel application note AP-723
*
* C-prototype:
*      void   CANinit(void);
_CANInit:
* init controller 82527
      LEA     (CAN).L,A0     controller base address

* set CPU interface register: (page 12)
*      SCLK = XTAL/2
*      MCLK = SCLK
*      disable CLKOUT signal
       MOVE.B  #$40,(2,A0)

* set CCE (Change Configuration Register)
* bit in Control register (page 9)
*      enable write access to configuration registers
*      prevent activities on the CAN-bus
       ORI.B   #$41,(A0)

* set bus configuration register (page 16)
*      bypass comparator
*      logical ones is recessive
*      disable TXl driver
*      DcR0 and DcR1 are don’t cares

       MOVE.B  #$48,($2f,A0)
```

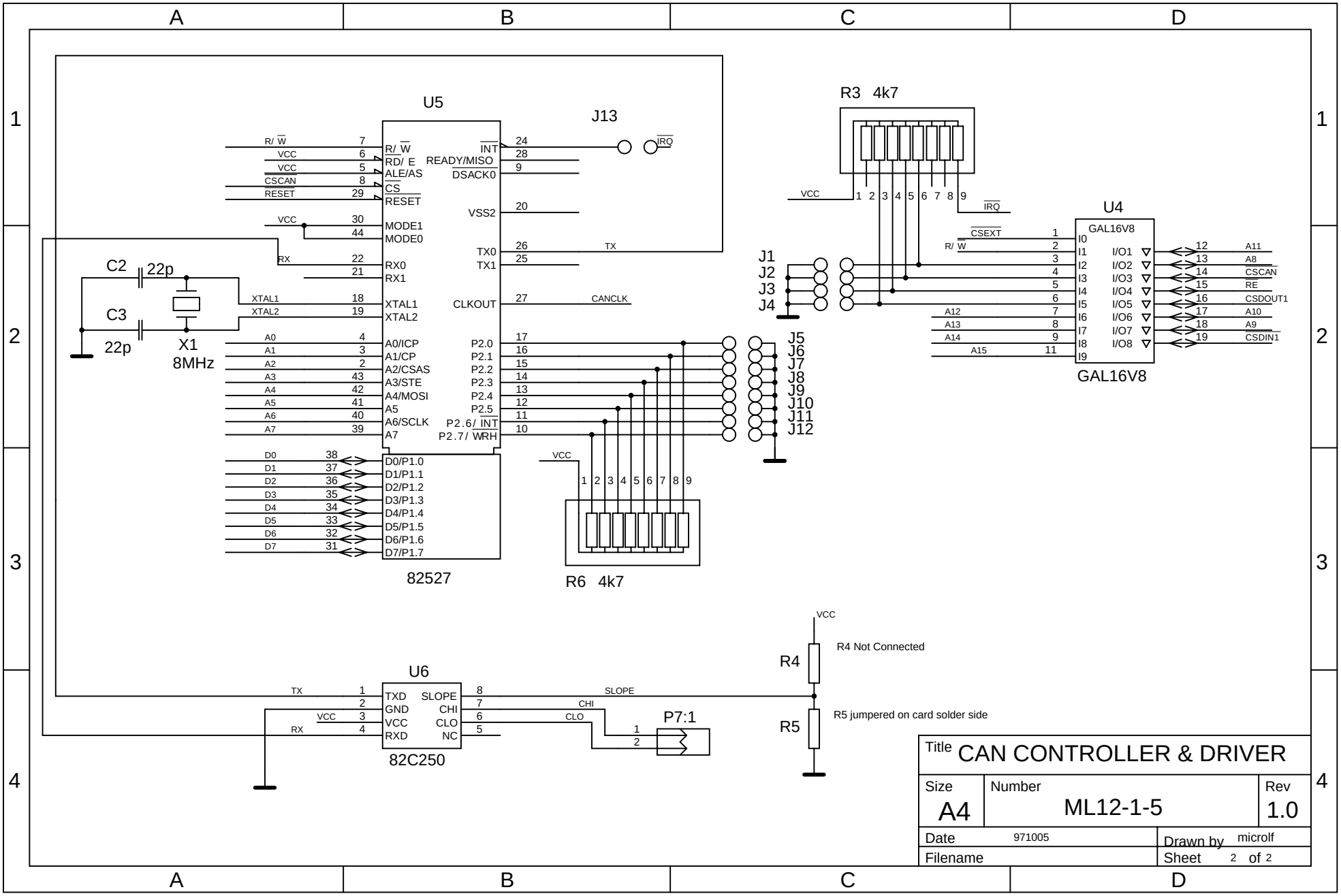
```
* set bit timing registers (page 18)
*      250 kBits/s
*      sampling mode is fast
*      TSEG1 = 6, TSEG2 = 7 (page 19)

       MOVE.B  #$40,($3f,A0)
       MOVE.B  #$67,($4f,A0)

* clear CCE bit
*      prevent write access of
*      configuration registers
       MOVE.B  #1,(A0)

* reset control register 0 and 1 for each message object
       MOVE.L  #$10,D0
resloop1:
       MOVE.B  #$55,(A0,D0)
       MOVE.B  #$55,(1,A0,D0)
       ADDI.L  #$10,D0
       CMPI.L  #$100,D0
       BNE      resloop1


* set global masks "don't care"
       CLR.B   (6,A0)
       CLR.B   (7,A0)
       CLR.B   (8,A0)
       CLR.B   (9,A0)
       CLR.B   ($a,A0)
       CLR.B   ($b,A0)


* load C8 into arbitration registers
* message object 1  (page 24)
* will set ID(0)
       MOVE.B  #$c8,($12,A0)
       MOVE.B  #$c8,($13,A0)
       MOVE.B  #$c8,($14,A0)
       MOVE.B  #$c8,($15,A0)


* Take chip out of the init mode
       CLR.B   (a0)
       RTS
```

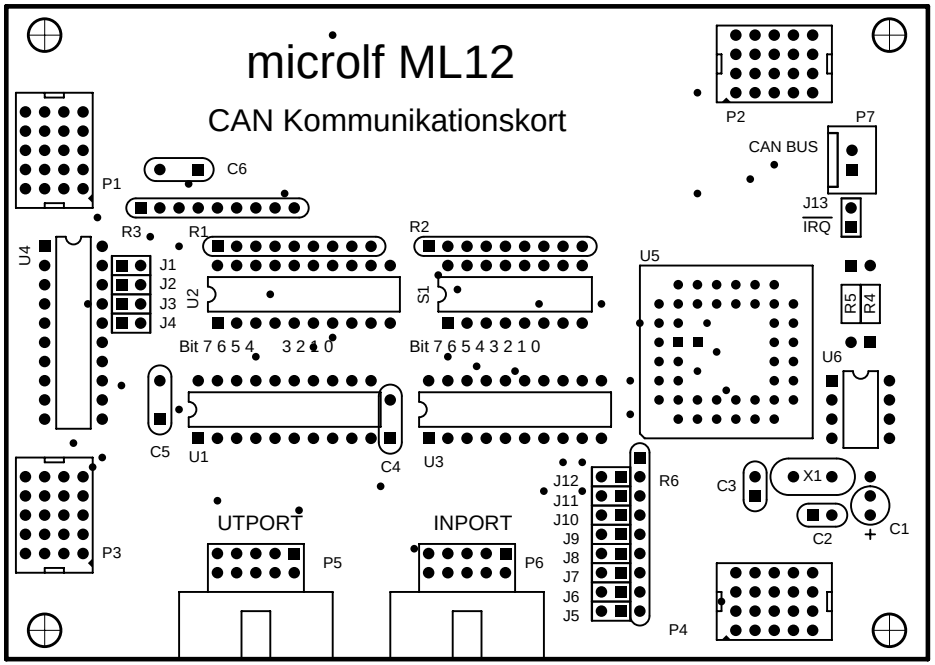
```
*********************************************
*       Simple  send routine for test purposes
*
* C-Prototype:
*       void CANSend(int msgobject, char *data);
_CANSend:
* calc base address for this message object
      MOVE.L  (4,SP),D0    get "msgobject"
      ROL.L   #4,D0
      LEA     (CAN).L,A0   base of controller
      ADDA.L  D0,A0        a0 holds address of control0
      MOVEA.L (8,SP),A1    address of data
* init for transmit
      MOVE.B  #$95,(A0)
      MOVE.B  #$59,(1,A0)
      MOVE.B  #$8c,(6,A0)
* update data start
      MOVE.B  #$FA,(1,A0)

* update data bytes
      CLR.L   D0              index
L2:
      MOVE.B  (A1)+,(7,A0,D0)
      ADDQ.L  #1,D0
      CMPI.L  #8,D0
      BNE     L2
* update data end
      MOVE.B  #$E7,(1,A0)

* return to caller
      RTS
```

```
**********************************
*       Set up message object as receiver
*       void CANSetupRec(int msgobj)
_CANSetupRec:
* calc base address for this message object
       MOVE.L  (4,SP),D0       get "msgobject"
       ROL.L   #4,D0
       LEA     (CAN).L,A0      base of controller
       ADDA.L  D0,A0           a0 holds address of
control0
* setup as receiver
       MOVE.B  #$95,(A0)
       MOVE.B  #$55,(1,A0)
       MOVE.B  #$4,(6,A0)
       RTS
*
* Non-blocking CAN-message receiver
* C-Prototype:
*      int    CANrec(int msgobject);
_CANRec:
* calc base address for this message object
       MOVE.L  (4,SP),D0       get "msgobject"
       ROL.L   #4,D0
       LEA     (CAN).L,A0      base of controller
       ADDA.L  D0,A0
       MOVEA.L (8,SP),A1       address of data
* see if message is pending
       BTST.B   #1,(1,A0)
       BEQ     RecRet0         no data

* Reset NewData
       MOVE.B  #$FD,(1,A0)
* pick up data
       CLR.L   D0              index
L4:
       MOVE.B  (7,A0,D0),(A1)+
       ADDQ.L  #1,D0
       CMPI.L  #8,D0
       BNE     L4
       MOVE.L  #1,D0           return TRUE
       RTS
RecRet0:
       CLR.L   D0              return FALSE
       RTS
```

```
/*
       File N1.C    (node 1, transmitter)
       C-test program
       MC68/ML12 - Can-controller
*/

/* prototypes for assembly routines */
void    CANInit(void);
void    CANSend(int , char *);

/* sample use */
char    Data[5][10];

void    main()
{
int     i,j;

       CANInit();      /* init hardware in this node */
       strcpy(Data[0],"Hej");
       strcpy(Data[1],"Du");
       strcpy(Data[2],"CAN..");
       j=0;
       while(1){
       /* Send the block */
              CANSend(1,Data[j]);
              for (i=0;i<100000;i++); /* wait a while */
              if(j==2)j=0; else j++;
              /* Show we are alive */
              puts("node 1 transmit");
       }
}
/*
       File N2.C    (node 2, receiver)
       MC68/ML12 - Can-controller
*/
/* prototypes for assembly routines */
void   CANInit(void);
int    CANRec(int , char *);
void   CANSetupRec(int);
char    Data[9];

void    main()
{
       CANInit();      /* init hardware in this node */
       CANSetupRec(1);
       while(1){
              /* Spin for a message block */
              if(CANRec(1,Data)){
              Data[8]='\0';
              puts(Data);
              }
       }
}
```


A
B
C
D
1
2
3
4
U1
RESET
CSDOUT1
CLR
CLK
1D 1Q
2D 2Q
3D 3Q
4D 4Q
5D 5Q
6D 6Q
7D 7Q
8D 8Q
D0
D1
D2
D3
D4
D5
D6
D7
DO0
DO1
DO2
DO3
DO4
DO5
DO6
DO7
74HC273
U2
R1
100
P5:1
GND
VCC
IDC2X5M
R2
4k7
U3
CSDIN1
GND
G
DIR
A1 B1
A2 B2
A3 B3
A4 B4
A5 B5
A6 B6
A7 B7
A8 B8
DI7
DI6
DI5
DI4
DI3
DI2
DI1
DI0
74HC245
S1
P6:1
IDC2X5M
P1:1
CSEXT
R/ W
AS
RESET
CLKOUT
TIN1
TOUT1
TGATE1
IDC2X5M
P2:1
D0
D1
D2
D3
D4
D5
D6
D7
IDC2X5M
P3:1
A15
A14
A13
A12
A11
A10
A9
A8
IDC2X5M
P4:1
A7
A6
A5
A4
A3
A2
A1
A0
IDC2X5M
VCC
C1
47u
C4
C5
C6
10n
10n
10n
Title
SIMPLE I/O and BUS
Size
A4
Number
ML12-1-1
Rev
1.0
Date
970910
Drawn by
microlf
Filename
Sheet
1
of
2


A
B
C
D
1
2
3
4
U5
82527
J13
IRQ
TX
CANCLK
C2
22p
C3
22p
X1
8MHz
XTAL1
XTAL2
RX
R/ W
VCC
CSCAN
RESET
MODE1
MODE0
RX0
RX1
TX0
TX1
VSS2
CLKOUT
INT
READY/MISO
DSACK0
A0/ICP
A1/CP
A2/CSAS
A3/STE
A4/MOSI
A5
A6/SCLK
A7
P2.0
P2.1
P2.2
P2.3
P2.4
P2.5
P2.6/ INT
P2.7/ WRH
D0/P1.0
D1/P1.1
D2/P1.2
D3/P1.3
D4/P1.4
D5/P1.5
D6/P1.6
D7/P1.7
R3 4k7
R6 4k7
J1
J2
J3
J4
J5
J6
J7
J8
J9
J10
J11
J12
U4
GAL16V8
CSEXT
I0
I1
I2
I3
I4
I5
I6
I7
I8
I9
I/O1
I/O2
I/O3
I/O4
I/O5
I/O6
I/O7
I/O8
A11
A8
CSCAN
RE
CSDOUT1
A10
A9
CSDIN1
A12
A13
A14
A15
U6
82C250
TXD
GND
VCC
RXD
SLOPE
CHI
CLO
NC
P7:1
R4
R5
R4 Not Connected
R5 jumpered on card solder side
Title
CAN CONTROLLER & DRIVER
Size
A4
Number
ML12-1-5
Rev
1.0
Date
971005
Drawn by
microlf
Filename
Sheet
2
of
2


microlf ML12
CAN Kommunikationskort
P1
C6
R3
R1
R2
P2
P7
CAN BUS
J13
IRQ
U4
J1
J2
J3
J4
U2
S1
Bit 7 6 5 4 3 2 1 0
Bit 7 6 5 4 3 2 1 0
U5
R5
R4
U6
C5
U1
C4
U3
J12
J11
J10
J9
J8
J7
J6
J5
R6
C3
X1
C2
C1
P3
UTPORT
INPORT
P5
P6
P4