Welcome to E-XFL.COM

What is "Embedded - Microcontrollers"?

"Embedded - Microcontrollers" refer to small, integrated circuits designed to perform specific tasks within larger systems. These microcontrollers are essentially compact computers on a single chip, containing a processor core, memory, and programmable input/output peripherals. They are called "embedded" because they are embedded within electronic devices to control various functions, rather than serving as standalone computers. Microcontrollers are crucial in modern electronics, providing the intelligence and control needed for a wide range of applications.

Applications of "Embedded - Microcontrollers"

Details	
Product Status	Obsolete
Core Processor	8051
Core Size	8-Bit
Speed	16MHz
Connectivity	CANbus, EBI/EMI, UART/USART
Peripherals	DMA, POR, PWM, WDT
Number of I/O	48
Program Memory Size	-
Program Memory Type	ROMless
EEPROM Size	-
RAM Size	512 x 8
Voltage - Supply (Vcc/Vdd)	4.5V ~ 5.5V
Data Converters	A/D 8x10b
Oscillator Type	Internal
Operating Temperature	-40°C ~ 85°C (TA)
Mounting Type	Surface Mount
Package / Case	80-BQFP
Supplier Device Package	80-PQFP (14x20)
Purchase URL	https://www.e-xfl.com/product-detail/nxp-semiconductors/p80ce598ffb-00-557

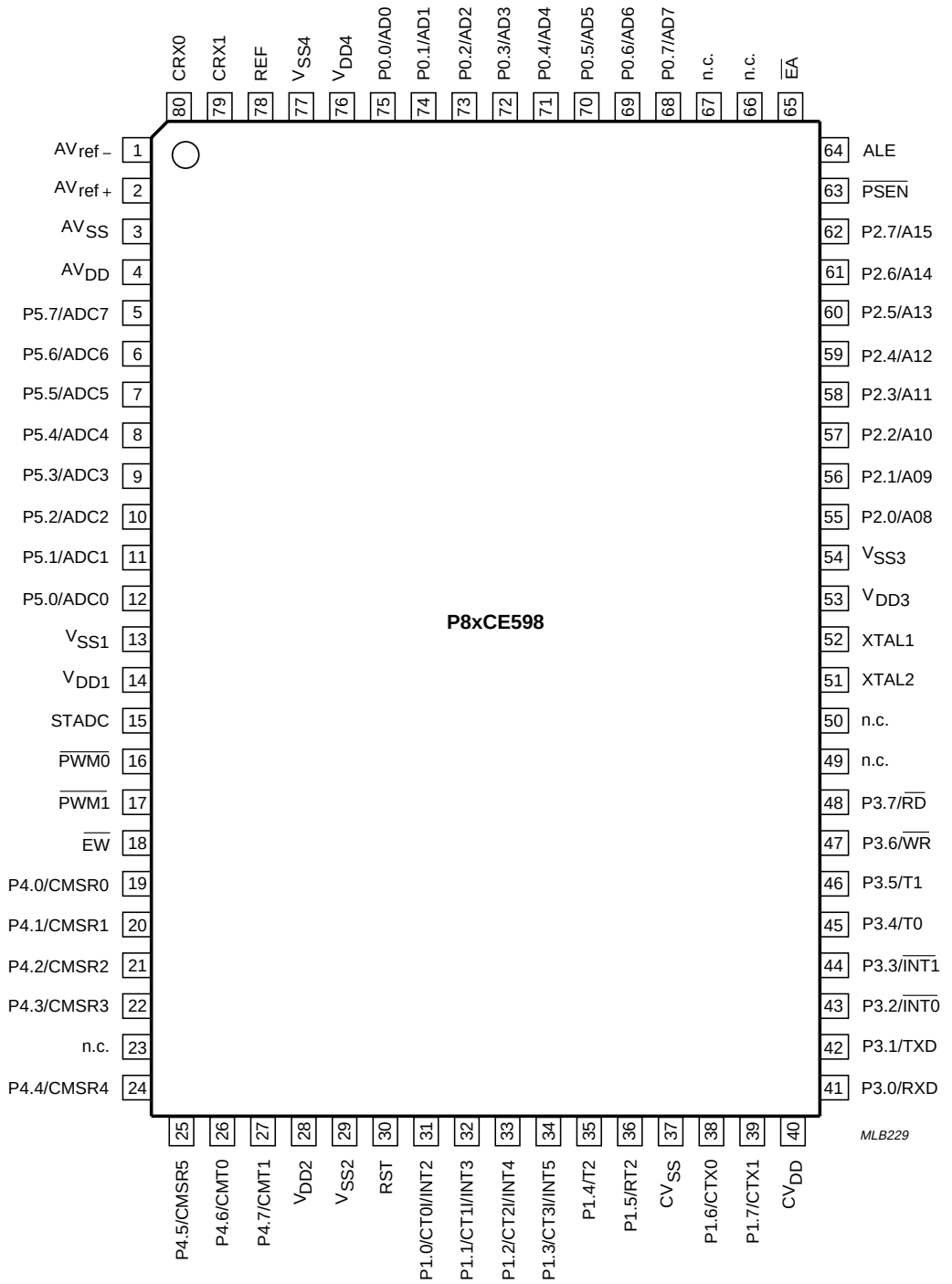


Fig.3 Pin configuration QFP80/SOT318-1.

SYMBOL		PIN	DESCRIPTION
DEFAULT	**ALTERNATIVE**		
Port 3			
P3.0 to P3.7		41 to 48	**8-bit quasi-bidirectional I/O port**.
	RXD	41	**Serial Input Port**.
	TXD	42	**Serial Output Port**.
	$\overline{\text{INT0}}$	43	**External interrupt input 0.**
	$\overline{\text{INT1}}$	44	**External interrupt input 1.**
	T0	45	**Timer 0 external input**.
	T1	46	**Timer 1 external input**.
	$\overline{\text{WR}}$	47	**External Data Memory Write strobe**.
	$\overline{\text{RD}}$	48	**External Data Memory Read strobe**.
Port 2 (Sink/source: 1 × TTL = 4 × LSTTL inputs)			
P2.0 to P2.7		55 to 62	**8-bit quasi-bidirectional I/O port**.
	A08 to A15		**High-order address byte for external memory**.
Port 0 (Sink/source: 8 × LSTTL inputs)			
P0.7 to P0.0		68 to 75	**8-bit open drain bidirectional I/O port**.
	AD7 to AD0		**Multiplexed Low-order address and Data bus for external memory**.
Port 5			
P5.7 to P5.0		5 to 12	**8-bit input port**.
	ADC7 to ADC0		**8 input channels to ADC**.

Notes

1. To avoid a ‘latch up’ effect at power-on: $V_{SS} - 0.5\ V <$ ‘voltage on any pin at any time’ $< V_{DD} + 0.5\ V$.
2. If the CAN-controller is in the reset state (e.g. after a power-up reset; CAN Control Register bit CR.0; see Section 13.5.3 Table 32, the CAN transmitter outputs are floating and the pins P1.6 and P1.7 can be used as open-drain port pins. After a power-up reset the port data is HIGH, leaving the pins P1.6 and P1.7 floating.

Table 30 CPU/CAN Register map

BIT							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Control Segment							
ADDRESS 0: CONTROL REGISTER							
TM	S	RA	OIE	EIE	TIE	RIE	RR
ADDRESS 1: COMMAND REGISTER							
RX0A	RX1A	WUM	SLP	COS	RRB	AT	TR
ADDRESS 2: STATUS REGISTER							
BS	ES	TS	RS	TCS	TBS	DO	RBS
ADDRESS 3: INTERRUPT REGISTER							
Reserved	Reserved	Reserved	WUI	OI	EI	TI	RI
ADDRESS 4: ACCEPTANCE CODE REGISTER							
AC.7	AC.6	AC.5	AC.4	AC.3	AC.2	AC.1	AC.0
ADDRESS 5: ACCEPTANCE MASK REGISTER							
AM.7	AM.6	AM.5	AM.4	AM.3	AM.2	AM.1	AM.0
ADDRESS 6: BUS TIMING REGISTER 0							
SJW.1	SJW.0	BRP.5	BRP.4	BRP.3	BRP.2	BRP.1	BRP.0
ADDRESS 7: BUS TIMING REGISTER 1							
SAM	TSEG2.2	TSEG2.1	TESG2.0	TSEG1.3	TSEG1.2	TSEG1.1	TSEG1.0
ADDRESS 8: OUTPUT CONTROL REGISTER							
OCTP1	OCTN1	OCPOL1	OCTP0	OCTN0	OCPOL0	OCMODE1	OCMODE0
ADDRESS 9: TEST REGISTER (note 1)							
Reserved	Reserved	Map Internal Register	Connect RX Buffer 0 CPU	Connect TX Buffer CPU	Access Internal Bus	Normal RAM Connect	Float Output Driver

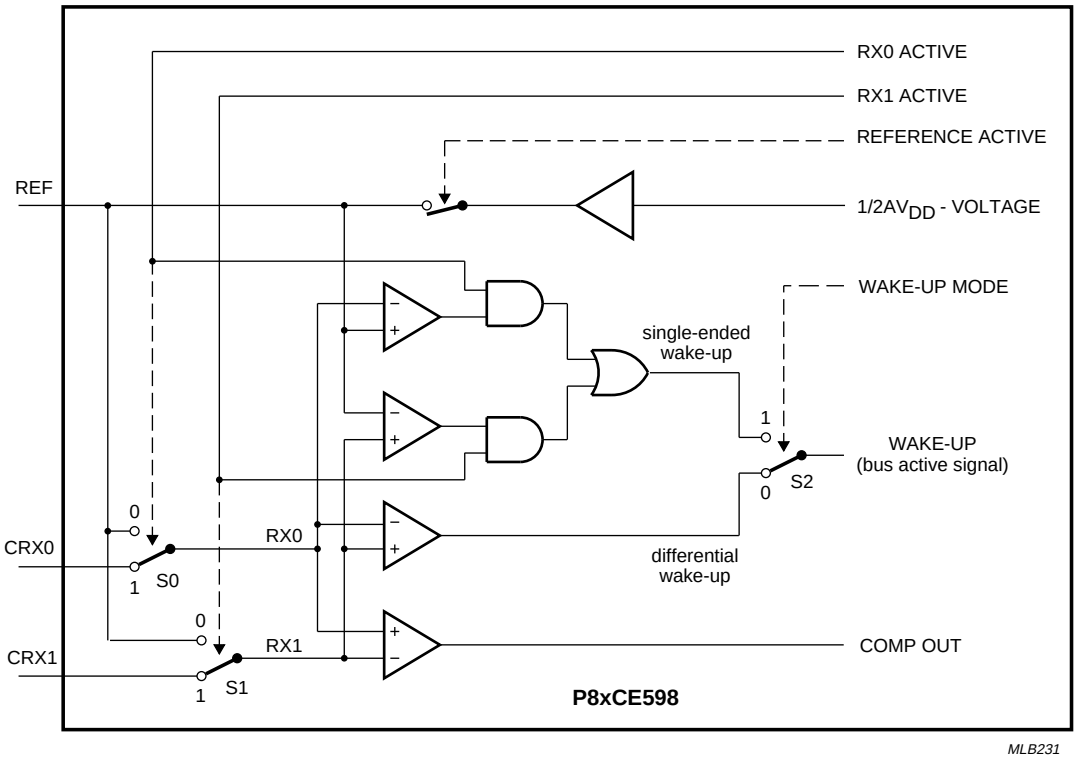


Fig.16 Configurable CAN receiver.

Notes to the description of the CMR bits

1. The RX0/RX1 Active bits, if being read, reflect the status of the respective switches (see Fig.16). It is recommended to change the switches only during the reset state (Reset Request = HIGH).
2. The Wake-Up Mode bit should be set at the same time as the Sleep bit. The differential wake up mode is useful if both bus wires are fully functioning; it minimizes the amount of wake ups due to noise. The single ended wake up mode is recommended if a wake up must be possible even if one bus wire is already or may become disturbed (see Fig.16).
3. The CAN-controller will enter sleep mode, if the Sleep bit is set HIGH (sleep) there is no bus activity and no interrupt is pending. The CAN-controller will wake up after the Sleep bit is set LOW (wake up) or when there is bus activity. On wake up, a Wake-Up Interrupt (see Section 13.5.6) is generated (see also Chapter 15). A CAN-controller which is sleeping and then awaken by bus activity will not be able to receive this message until it detects a Bus-Free signal (see Section 13.6.9.6). The Sleep bit, if read, reflects the status of the CAN-controller.
4. This command bit is used to acknowledge the Data Overrun condition signalled by the Data Overrun status bit. Command is given only after releasing both receive buffers. The stored messages have to be rejected. The command bit is set simultaneously with setting of the Release Receive Buffer command bit the second time.
5. After reading the contents of the Receive Buffer (RBF0 or RBF1) the CPU must release this buffer by setting Release Receive Buffer bit HIGH (released). This may result in another message becoming immediately available.
   To prevent the RRB command being executed only once, the minimum wait time between two successive RRB commands is 3 system clock cycles (t_{SCL}, see Section 13.5.9).
6. The Abort Transmission bit is used when the CPU requires the suspension of the previously requested transmission, e.g. to transmit an urgent message. A transmission already in progress is not stopped. In order to see if the original message had been either transmitted successfully or aborted, the Transmission Complete Status bit should be checked. This should be done after the Transmit Buffer Access bit has been set HIGH (released) or a Transmit Interrupt has been generated (see Section 13.5.6).
7. If the Transmission Request bit was set HIGH in a previous command, it cannot be cancelled by setting the Transmission Request bit LOW (absent). Cancellation of the requested transmission may be performed by setting the Abort Transmission bit HIGH (present).

Table 35 Combination of bits RX0A and RX1A (see Fig.16)

CONTROL		RX0	RX1
RX0ACTIVE	**RX1ACTIVE**		
1	1	CRX0	CRX1
1	0	CRX0	$\frac{1}{2}AV_{DD}$
0	1	$\frac{1}{2}AV_{DD}$	CRX1
0	0	No action	

Table 40 Effects of setting the Reset Request bit HIGH (present)

TYPE	BIT	SYMBOL	FUNCTION	EFFECT
Control	CR.7	TM	Test Mode	LOW (disabled)
	CR.5	RA	Reference Active	HIGH (output); note 1
Command	CMR.7	RX0A	RX0 Active	HIGH (RX0 = CRX0); note 1
	CMR.6	RX1A	RX1 Active	HIGH (RX1 = CRX1); note 1
	CMR.4	SLP	Sleep	LOW (wake-up)
	CMR.3	COS	Clear Overrun Status	HIGH (clear)
	CMR.2	RRB	Release Receive Buffer	HIGH (released)
	CMR.1	AT	Abort Transmission	LOW (absent)
	CMR.0	TR	Transmission Request	LOW (absent)
Status	SR.7	BS	Bus Status	LOW (Bus-On); note 1
	SR.6	ES	Error Status	LOW (no error); note 1
	SR.5	TS	Transmit Status	LOW (idle)
	SR.4	RS	Receive Status	LOW (idle)
	SR.3	TCS	Transmission Complete Status	HIGH (complete)
	SR.2	TBS	Transmit Buffer Access	HIGH (released)
	SR.1	DO	Data Overrun	LOW (absent)
	SR.0	RBS	Receive Buffer Status	LOW (empty)
Interrupt	IR.3	OI	Overrun Interrupt	LOW (reset)
	IR.1	TI	Transmit Interrupt	LOW (reset)
	IR.0	RI	Receive Interrupt	LOW (reset)

Note

1. Only after an external reset; see note 5 to Table 37 “Description of the SR bits”.

13.5.7 Acceptance Code Register (ACR)

The Acceptance Code Register is part of the acceptance filter of the CAN-controller. This register can be accessed (read/write), if the Reset Request bit is set HIGH (present).

When a message is received which passes the acceptance test and if there is an empty Receive Buffer, then the respective Descriptor and Data Field (see Fig.15) are sequentially stored in this empty buffer.

In the event that there is no empty Receive Buffer, the Data Overrun bit is set HIGH (overrun); see Sections 13.5.5 and 13.5.6.

When the complete message has been correctly received the following occurs:

- The Receive Buffer Status bit is set HIGH (full)
- If the Receive Interrupt Enable bit is set HIGH (enabled), the Receive Interrupt is set HIGH (set).

During transmission of a message which passes the acceptance test, the message is also written to its own Receive Buffer. If no Receiver Buffer is available, Data Overrun is signalled because it is not known at the start of a message whether the CAN-controller will lose arbitration and so become a receiver of the message.

Table 41 Acceptance Code Register (address 4)

7	6	5	4	3	2	1	0
AC.7	AC.6	AC.5	AC.4	AC.3	AC.2	AC.1	AC.0

Table 42 Description of the ACR bits

BIT	SYMBOL	FUNCTION
7 to 0	AC.7 to AC.0	**Acceptance Code.** The Acceptance Code bits (AC.7 to AC.0) and the eight most significant bits of the message's Identifier (ID.10 to ID.3) must be equal to those bit positions which are marked relevant by the Acceptance Mask bits (AM.7 to AM.0). The acceptance is given, if the following equation is satisfied: (ID10 ... ID.3) = [(AC.7 ... AC.0) or (AM.7 ... AM.0)] = 1111 1111 B

13.5.8 Acceptance Mask Register (AMR)

The Acceptance Mask Register is part of the acceptance filter of the CAN-controller.
This register can be accessed (read/write) if the Reset Request bit is set HIGH (present).

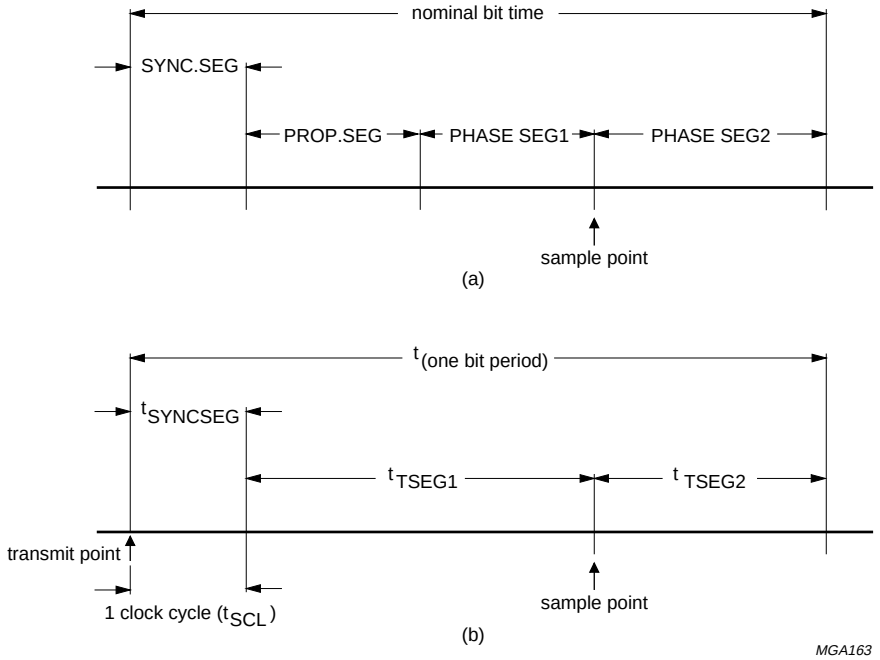
The Acceptance Mask Register qualifies which of the corresponding bits of the acceptance code are ‘relevant’ or ‘don't care’ for acceptance filtering.

Table 43 Acceptance Mask Register (address 5)

7	6	5	4	3	2	1	0
AM.7	AM.6	AM.5	AM.4	AM.3	AM.2	AM.1	AM.0

Table 44 Description of the AMR bits

BIT	SYMBOL	FUNCTION
7 to 0	AM.7 to AM.0	**Acceptance Mask.** If the Acceptance Mask bit is: HIGH (don’t care), then this bit position is ‘don’t care’ for the acceptance of a message. LOW (relevant), then this bit position is ‘relevant’ for acceptance filtering.



(a) As defined by the CAN-protocol.
(b) As implemented in the P8xCE598's on-chip CAN-controller.

Fig.18 Bit period.

13.5.19.2 Time Segment 1 (TSEG1)

This segment determines the location of the sampling point within a bit period, which is at the end of TSEG1. TSEG1 is programmable from 1 to 16 system clock cycles (see Section 13.5.10).

The correct location of the sample point is essential for the correct functioning of a transmission. The following points must be taken into consideration:

- A Start-Of-Frame (see Section 13.6.2) causes all CAN-controllers to perform a 'hard synchronization' (see Section 13.5.20) on the first recessive-to-dominant edge. During arbitration, however, several CAN-controllers may simultaneously transmit. Therefore it may require twice the sum of bus-line, input comparator and the output driver delay times until the bus is stable. This is the propagation delay time.
- To avoid sampling at an incorrect position, it is necessary to include an additional synchronization buffer on both sides of the sample point. The main reasons for incorrect sampling are:
  - incorrect synchronization due to spikes on the bus-line
  - slight variations in the oscillator frequency of each CAN-controller in the network, which results in a phase error.
- Time Segment 1 consists of the segment for compensation of propagation delays and the synchronization buffer segment directly before the sample point (see Fig.18).

13.5.19.3 Time Segment 2 (TSEG2)

This time segment provides:

- additional time at the sample point for calculation of the subsequent bit levels (e.g. arbitration)
- synchronization buffer segment directly after the sample point.

TSEG2 is programmable from 1 to 8 system clock cycles (see Section 13.5.10.

13.5.19.4 Synchronisation Jump Width (SJW)

SJW defines the maximum number of clock cycles (t_{SCL}) a period may be reduced or increased by one resynchronization. SJW is programmable from 1 to 4 system clock cycles, see Section 13.5.2.

13.5.19.5 Propagation Delay Time (t_{prop})

The Propagation Delay Time is:

$$t_{prop} = 2 \times (\text{physical bus delay} + \text{input comparator delay} + \text{output driver delay}).$$

t_{prop} is rounded up to the nearest multiple of t_{SCL}.

13.5.19.6 Bit Timing Restrictions

Restrictions on the configuration of the bit timing are based on internal processing. The restrictions are:

- $t_{TSEG2} \geq 2t_{SCL}$
- $t_{TSEG2} \geq t_{SJW}$
- $t_{TSEG1} \geq t_{SEG2}$
- $t_{TSEG1} \geq t_{SJW} + t_{prop}$.

The three sample mode (SAM = HIGH) has the effect of introducing a delay of one system clock cycle on the bus-line. This must be taken into account for the correct calculation of TSEG1 and TSEG2:

- $t_{TSEG1} \geq t_{SJW} + t_{prop} + 2t_{SCL}$
- $t_{TSEG2} \geq 3t_{SCL}$.

13.5.20 SYNCHRONIZATION

Synchronization is performed by a state machine which compares the incoming edge with its actual bit timing and adapts the bit timing by hard synchronization or resynchronization.

This type of synchronization occurs only at the beginning of a message.

The CAN-controller synchronizes on the first incoming recessive-to-dominant edge of a message (being the leading edge of a message's Start-Of-Frame bit; see Section 13.6.2.

Resynchronization occurs during the transmission of a message's bit stream to compensate for:

- Variations in individual CAN-controller oscillator frequencies
- Changes introduced by switching from one transmitter to another (e.g. during arbitration).

As a result of resynchronization either t_{TSEG1} may be increased by up to a maximum of t_{SJW} or t_{TSEG2} may be decreased by up to a maximum of t_{SJW}:

- $t_{TSEG1} \leq t_{SCL}$ [(TSEG1 + 1) + (SJW + 1)]
- $t_{TSEG2} \geq t_{SCL}$ [(TSEG2 + 1) – (SJW + 1)].

TSEG1, TSEG2 and SJW are the programmed numerical values.

The phase error (e) of an edge is given by the position of the edge relative to SYNCSEG, measured in system clock cycles (t_{SCL}).

The value of the phase error is defined as:

- e = 0, if the edge occurs within SYNCSEG
- e > 0, if the edge occurs within TSEG1
- e < 0, if the edge occurs within TSEG2.

The effect of resynchronization is:

- The same as that of a hard synchronization, if the magnitude of the phase error (e) is less or equal to the programmed value of t_{SJW}
- To increase a bit period by the amount of t_{SJW},if the phase error is positive and the magnitude of the phase error is larger than t_{SJW}
- To decrease a bit period by the amount of t_{SJW} if the phase error is negative and the magnitude of the phase error is larger than t_{SJW}.

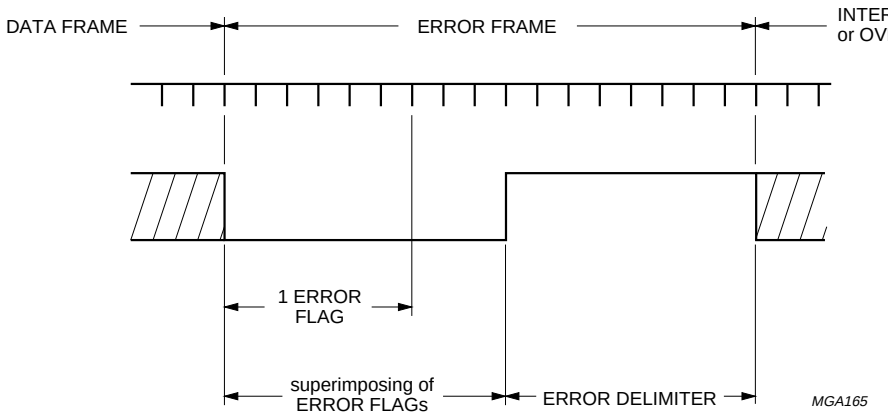


Fig.20 Error Frame.

13.6.5 Overload Frame

The Overload Frame consists of two fields:

- The Overload Flag
- The Overload Delimiter.

The transmission of an Overload Frame may only start:

- Condition 1; during the first bit period of an expected Intermission Field.
- Condition 2; one bit period after detecting the dominant bit during Intermission Field.

The P8xCE598's on-chip CAN-controller will never initiate transmission of a condition 1 Overload Frame and will only react on a transmitted condition 2 Overload Frame, according to the CAN-protocol. No more than two Overload Frames are generated to delay a Data Frame or a Remote Frame. Although the overall form of the Overload Frame corresponds to that of the Error Frame, an Overload Frame does not initiate or require the retransmission of the preceding frame.

13.6.5.1 Overload Flag

The Overload Flag consists of six dominant bits and has a similar format to the Error Flag.

There are two conditions in the CAN-protocol which lead to the transmission of an Overload Flag:

- Condition 1; receiver circuitry requires more time to process the current data before receiving the next frame (receiver not ready).
- Condition 2; detection of a dominant bit during Intermission Field (see Section 13.6.6).

The Overload Flag's form corrupts the fixed form of the Intermission Field. All other CAN-controllers detecting the overload condition also transmit an Overload Flag (condition 2).

13.6.5.2 Overload Delimiter

The Overload Delimiter consists of eight recessive bits and takes the same form as the Error Delimiter. After transmission of an Overload Flag, each CAN-controller monitors the bus-line until it detects a transition from a dominant-to-recessive bit level. At this point in time, every CAN-controller has finished sending its Overload Flag and all CAN-controllers start simultaneously transmitting seven more recessive bits.

13.6.6 INTER-FRAME SPACE

Data Frames and Remote Frames are separated from preceding frames (all types) by an Inter-Frame Space, consisting of an Intermission Field and a Bus-Idle. Error-passive CAN-controllers also send a Suspend Transmission (see Section 13.6.9) after transmission of a message. Overload Frames and Error Frames are not preceded by an Inter-Frame Space.

13.6.6.1 Intermission Field

The Intermission Field consists of three recessive bits. During an Intermission period, no frame transmissions will be started by the P8xCE598's on-chip CAN-controller. An Intermission is required to have a fixed time period to allow a CAN-controller to execute internal processes prior to the next receive or transmit task.

13.6.6.2 Bus-Idle

The Bus-Idle time may be of arbitrary length (min. 0 bit). The bus is recognized to be free and a CAN-controller having information to transmit may access the bus. The detection of a dominant bit level during Bus-Idle on the bus is interpreted as the Start-Of-Frame.

13.6.7 BUS ORGANIZATION

Bus organization is based on five basic rules described in the following subsections.

13.6.7.1 Bus Access

CAN-controllers only start transmission during the Bus-Idle state. All CAN-controllers synchronize on the leading edge of the Start-Of-Frame (hard synchronization).

13.6.7.2 Bus Arbitration

If two or more CAN-controllers simultaneously start transmitting, the bus access conflict is solved by a bit-wise arbitration process during transmission of the Arbitration Field.

During arbitration every transmitting CAN-controller compares its transmitted bit level with the monitored bus level. Any CAN-controller which transmits a recessive bit and monitors a dominant bus level immediately becomes the receiver of the higher-priority message on the bus without corrupting any information on the bus. Each message contains an unique Identifier and a RTR bit describing the type of data within the message. The Identifier together with the RTR bit implicitly define the message's bus access priority. During arbitration the most significant bit of the Identifier is transmitted first and the RTR bit last. The message with the lowest binary value of the Identifier and RTR bit has the highest priority. A Data Frame has higher priority than a Remote Frame due to its RTR bit having a dominant level.

For every Data Frame there is an unique transmitter. For reasons of compatibility with other CAN-bus controllers, use of the Identifier bit pattern ID = 1111111XXXXB (X being bits of arbitrary level) is forbidden.

The number of available different Identifiers: $(2^{11} - 2^4) = 2032$.

13.6.7.3 Coding/Decoding

The following bit fields are coded using the bit-stuffing technique:

- Start-Of-Frame
- Arbitration Field
- Control Field
- Data Field
- CRC Sequence.

When a transmitting CAN-controller detects five consecutive bits of identical polarity to be transmitted, a complementary (stuff) bit is inserted into the transmitted bit-stream.

When a receiving CAN-controller has monitored five consecutive bits with identical polarity in the received bit streams of the above described bit fields, it automatically deletes the next received (stuff) bit. The level of the deleted stuff bit has to be the complement of the previous bits; otherwise a Stuff Error will be detected and signalled (see Section 13.6.8).

The remaining bit fields or frames are of fixed form and are not coded or decoded by the method of bit-stuffing.

The bit-stream in a message is coded according to the Non-Return-to-Zero (NRZ) method, i.e. during a bit period, the bit level is held constant, either recessive or dominant.

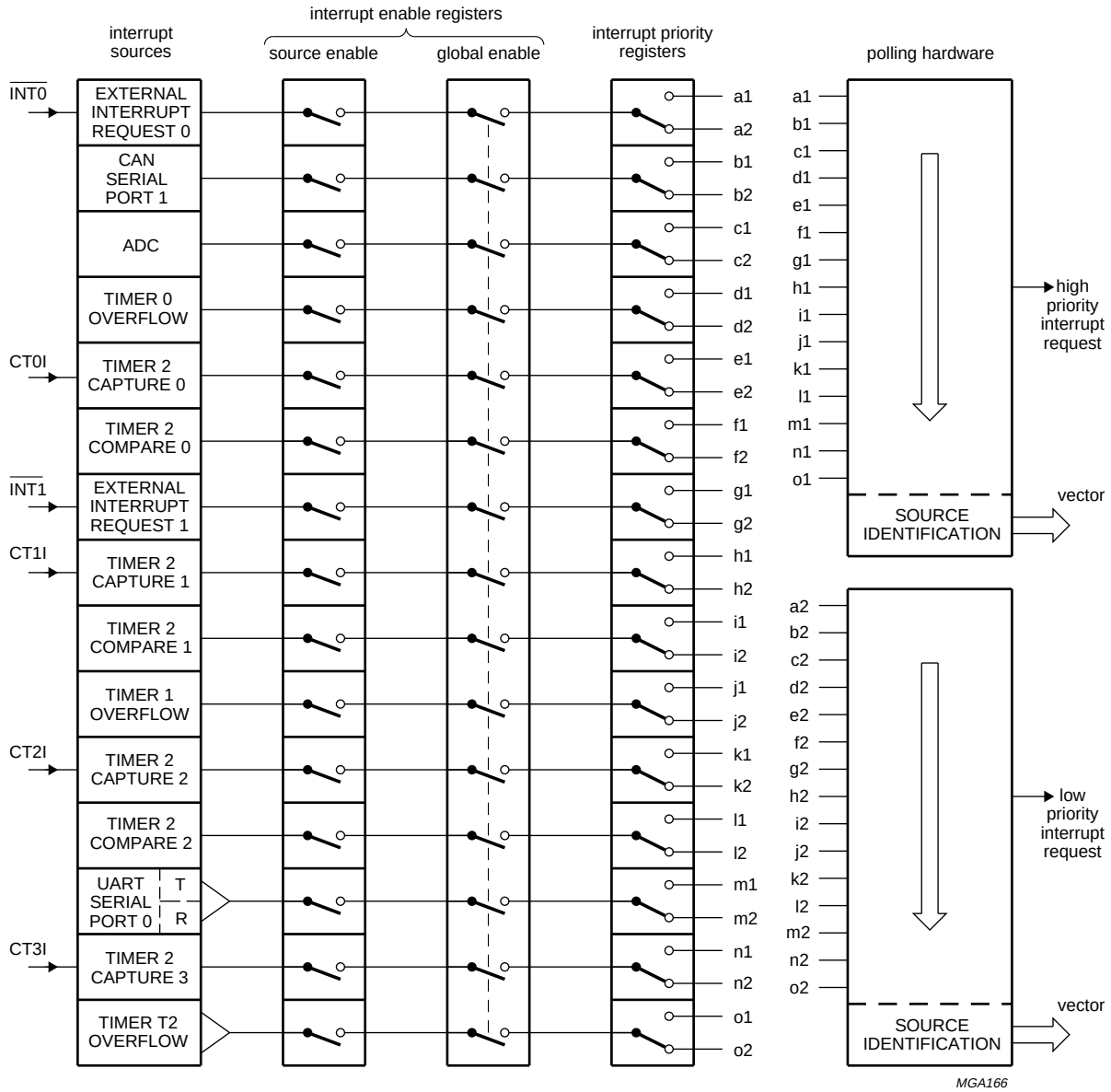


Fig.21 Interrupt system.

14.1 Interrupt Enable and Priority Registers

14.1.1 Interrupt Enable register 0 (IEN0)

Table 71 Interrupt Enable register 0 (address A8H)

7	6	5	4	3	2	1	0
EA	EAD	ES1	ES0	ET1	EX1	ET0	EX0

Table 72 Description of the IEN0 bits

BIT	SYMBOL	FUNCTION
7	EA	**General enable/disable control**. If bit EA is: LOW, then no interrupt is enabled. HIGH, then any individually enabled interrupt will be accepted.
6	EAD	Enable ADC interrupt.
5	ES1	Enable SIO1 (CAN) interrupt.
4	ES0	Enable SIO0 (UART) interrupt.
3	ET1	Enable Timer 1 interrupt.
2	EX1	Enable External 1 interrupt.
1	ET0	Enable Timer 0 interrupt.
0	EX0	Enable External 0 interrupt.

14.1.2 Interrupt Enable register 1 (IEN1)

Table 73 Interrupt Enable register 0 (address E8H)

7	6	5	4	3	2	1	0
ET2	ECM2	ECM1	ECM0	ECT3	ECT2	ECT1	ECT0

Table 74 Description of the IEN1 bits

Logic 0 = interrupt disabled; logic 1 = interrupt enabled.

BIT	SYMBOL	FUNCTION
7	ET2	Enable T2 overflow interrupt(s).
6	ECM2	Enable T2 comparator 2 interrupt.
5	ECM1	Enable T2 comparator 1 interrupt.
4	ECM0	Enable T2 comparator 0 interrupt.
3	ECT3	Enable T2 capture register 3 interrupt.
2	ECT1	Enable T2 capture register 2 interrupt.
1	ECT1	Enable T2 capture register 1 interrupt.
0	ECT0	Enable T2 capture register 0 interrupt.

16 OSCILLATOR CIRCUITRY

The oscillator circuitry of the P8xCE598 is a single-stage inverting amplifier in a Pierce oscillator configuration. The circuitry between XTAL1 and XTAL2 is basically an inverter biased to the transfer point. Either a crystal or ceramic resonator can be used as the feedback element to complete the oscillator circuitry. Both are operated in parallel resonance. XTAL1 (pin 52) is the high gain amplifier input, and XTAL2 (pin 51) is the output (see Fig.23). If XTAL1 is driven from an external source, XTAL2 must be left open (see Fig.24).

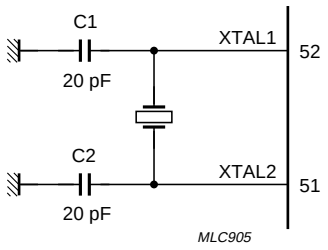


Fig.23 P8xCE598 oscillator circuit.

17 RESET CIRCUITRY

The reset pin RST is connected to a Schmitt trigger for noise rejection (see Fig.25). A reset is accomplished by holding the RST pin HIGH for at least two machine cycles (24 oscillator periods). The CPU responds by executing an internal reset. During reset ALE and $\overline{PSEN}$ output a HIGH level. In order to perform a correct reset, this level must not be affected by external elements.

Also with the P8xCE598, the RST line can be pulled HIGH internally by a pull-up transistor activated by the Watchdog timer T3. The length of the output pulse from T3 is 3 machine cycles. A pulse of such short duration is necessary in order to recover from a processor or system fault as fast as possible.

During Power-down a reset could be generated internally via the CAN Wake-Up interrupt. Then the RST pin is pulled HIGH for 6144 machine cycles. In this case the CAN Controller is not reset.

If the Watchdog timer or the CAN Wake-Up interrupt is used to reset external devices, the usual capacitor arrangement for Power-on-reset (see Fig.26) should not be used.

However, the internal reset is forced, independent of the external level on the RST pin.

The MAIN RAM and AUXILIARY RAM are not affected. When V_{DD} is turned on, the RAM content is indeterminate. A reset leaves the internal registers as shown in Table 83).

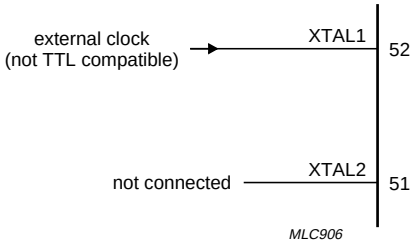


Fig.24 Driving P8xCE598 from an external source.

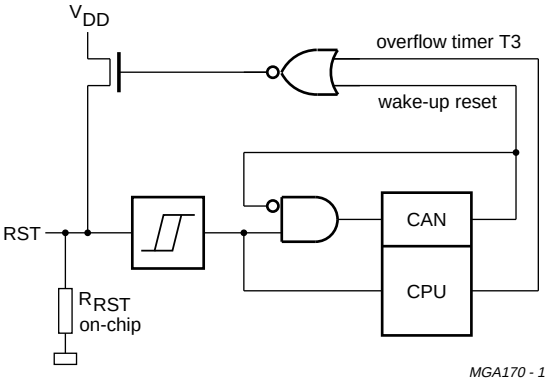


Fig.25 On-chip reset configuration.

Table 88 Description of the mnemonics in the Instruction set

MNEMONIC	DESCRIPTION
Data addressing modes	
Rr	Working register R0-R7.
direct	128 internal RAM locations and any special function register (SFR).
@Ri	Indirect internal RAM location addressed by register R0 or R1 of the actual register bank.
#data	8-bit constant included in instruction.
#data 16	16-bit constant included as bytes 2 and 3 of instruction.
bit	Direct addressed bit in internal RAM or SFR.
addr16	16-bit destination address. Used by LCALL and LJMP. The branch will be anywhere within the 64 kbytes Program Memory address space.
addr11	11-bit destination address. Used by ACALL and AJMP. The branch will be within the same 2 kbytes page of Program Memory as the first byte of the following instruction.
rel	Signed (two's complement) 8-bit offset byte. Used by SJMP and all conditional jumps. Range is –128 to +127 bytes relative to first byte of the following instruction.
Hexadecimal opcode cross-reference	
*	8, 9, A, B, C, D, E, F.
•	1, 3, 5, 7, 9, B, D, F.
♦	0, 2, 4, 6, 8, A, C, E.

Table 89 Instruction map

First hexadecimal character of opcode ↓	← Second hexadecimal character of opcode →															
	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	NOP	AJMP addr11	LJMP addr16	RR A	INC A	INC direct	INC @Ri		INC Rr							
							0	1	0	1	2	3	4	5	6	7
1	JBC bit,rel	ACALL addr11	LCALL addr16	RRC A	DEC A	DEC direct	DEC @Ri		DEC Rr							
							0	1	0	1	2	3	4	5	6	7
2	JB bit,rel	AJMP addr11	RET	RL A	ADD A,#data	ADD A,direct	ADD A,@Ri		ADD A,Rr							
							0	1	0	1	2	3	4	5	6	7
3	JNB bit,rel	ACALL addr11	RETI	RLC A	ADDC A,#data	ADDC A,direct	ADDC A,@Ri		ADDC A,Rr							
							0	1	0	1	2	3	4	5	6	7
4	JC rel	AJMP addr11	ORL direct,A	ORL direct,#data	ORL A,#data	ORL A,direct	ORL A,@Ri		ORL A,Rr							
							0	1	0	1	2	3	4	5	6	7
5	JNC rel	ACALL addr11	ANL direct,A	ANL direct,#data	ANL A,#data	ANL A,direct	ANL A,@Ri		ANL A,Rr							
							0	1	0	1	2	3	4	5	6	7
6	JZ rel	AJMP addr11	XRL direct,A	XRL direct,#data	XRL A,#data	XRL A,direct	XRL A,@Ri		XRL A,Rr							
							0	1	0	1	2	3	4	5	6	7
7	JNZ rel	ACALL addr11	ORL C,bit	JMP @A+DPTR	MOV A,#data	MOV direct,#data	MOV @Ri,#data		MOV Rr,#data							
							0	1	0	1	2	3	4	5	6	7
8	SJMP rel	AJMP addr11	ANL C,bit	MOVC A,@A+PC	DIV AB	MOV direct,direct	MOV direct,@Ri		MOV direct,Rr							
							0	1	0	1	2	3	4	5	6	7
9	MOV DTPR,#data16	ACALL addr11	MOV bit,C	MOVC A,@A+DPTR	SUBB A,#data	SUBB A,direct	SUBB A,@Ri		SUB A,Rr							
							0	1	0	1	2	3	4	5	6	7
A	ORL C,/bit	AJMP addr11	MOV bit,C	INC DPTR	MUL AB		MOV @Ri,direct		MOV Rr,direct							
							0	1	0	1	2	3	4	5	6	7
B	ANL C,/bit	ACALL addr11	CPL bit	CPL C	CJNE A,#data,rel	CJNE A,direct,rel	CJNE @Ri,#data,rel		CJNE Rr,#data,rel							
							0	1	0	1	2	3	4	5	6	7
C	PUSH direct	AJMP addr11	CLR bit	CLR C	SWAP A	XCH A,direct	XCH A,@Ri		XCH A,Rr							
							0	1	0	1	2	3	4	5	6	7
D	POP direct	ACALL addr11	SETB bit	SETB C	DA A	DJNZ direct,rel	XCHD A,@Ri		DJNZ Rr,rel							
							0	1	0	1	2	3	4	5	6	7
E	MOVX A,@DTPR	AJMP addr11	MOVX A,@Ri		CLR A	MOV A,direct [1]	MOV A,@Ri		MOV A,Rr							
			0	1			0	1	0	1	2	3	4	5	6	7
F	MOVX @DTPR,A	ACALL addr11	MOVX @Ri,A		CPL A	MOV direct,A	MOV @Ri,A		MOV Rr,A							
			0	1			0	1	0	1	2	3	4	5	6	7

Note

1. MOV A, ACC is not a valid instruction.

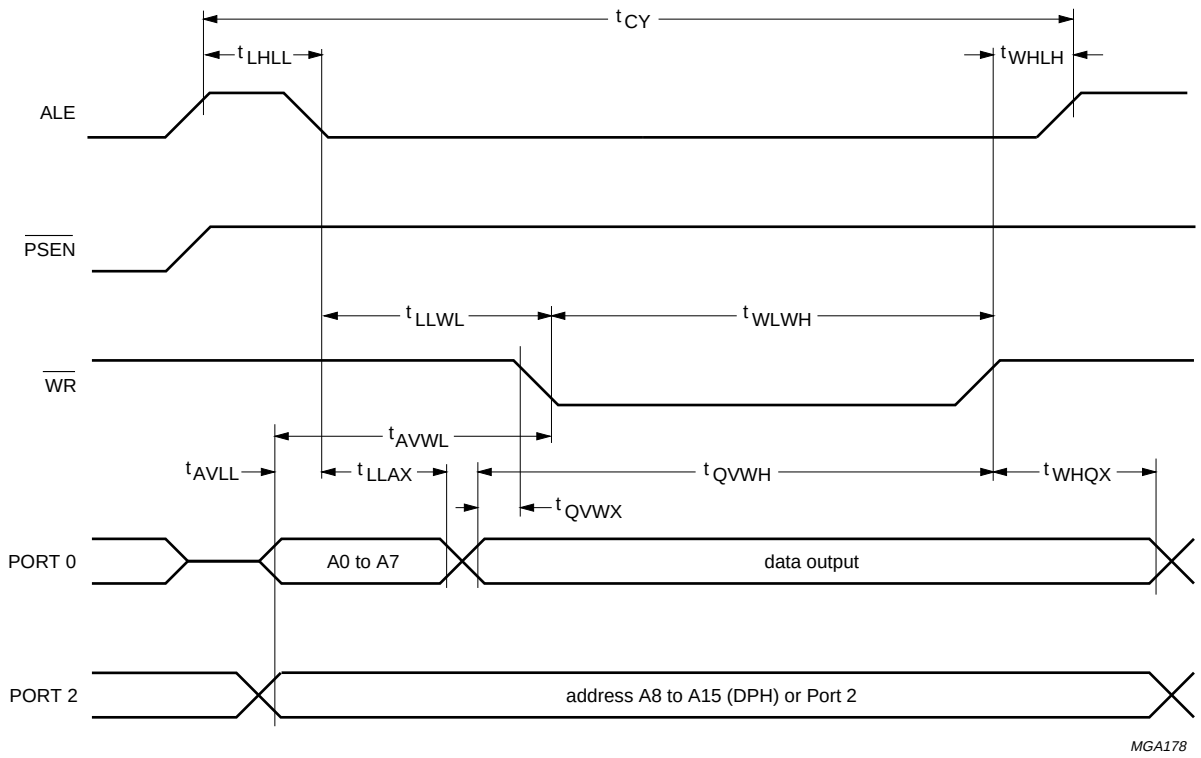


Fig.33 Write to external Data Memory.

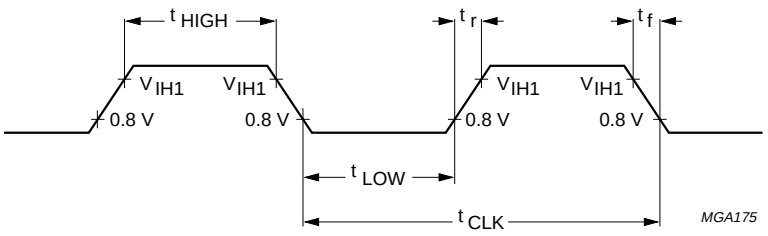


Fig.34 External clock drive XTAL1 (see Table 91).

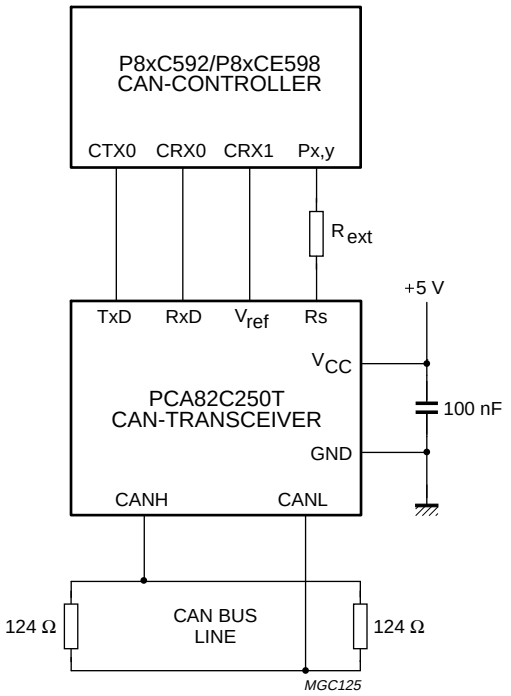


Fig.36 Application of a CAN-Transceiver (PCA82C250T).

22.2.5 P8xCE598 CAN Interrupt Handler Software example (including fast DMA transfer)

MCS-51 MACRO ASSEMBLER CAN interrupt-handler

```
LOC  OBJ   LINE  SOURCE

           1     $TITLE 8XCE598 (CAN interrupt-handler)
00A0       2     $NOSYMBOLS NOPAGING
00A1       3
           4     ;*****************************************************************************************************
           5     ;
           6     ;Very fast receive-routine for the 8XCE598 CAN Interface. It:
           7     • is embedded in the interrupt-handler for the CAN Controller,
           8     • uses the DMA-logic and
           9     • handles up to eight different messages
00A2       10    ;(if these have the same leading 8 identifier-bits).
           11    ;
           12    ;To allow for faster receive-routine, it is assumed that all other routines
           13    ;accessing the CAN Controller, disable the interrupt of the CAN Controller
           14    ;(IEN0.5) during their execution.
00A5       15    ;
00A7       16    ;Version:    1.0
           17    ;Date:       12-April-90
           18    ;Author:     Bernhard Reckels
           19    ;at:         Philips Components Application Lab., Hamburg (PCALH)
00A9       20
00AB       21    ;*****************************************************************************************************
00AD       22
           23    ;*****************************************************************************************************
           24    ;initial stuff
           25    ;*****************************************************************************************************
           26
           27    ;equatas
           28
           29              ;addresses of Special Function Registers
00AE       30              CANADR      EQU     0DBH
00AF       31              CANDAT      EQU     0DAH
           32              CANCON      EQU     0D9H
00B0       33              CANSTA      EQU     0D8H
           34
```

```
LOC   OBJ      LINE  SOURCE
00A0           107          ; determine the destination address in data-memory for the
00A1           108          ; message's Data-Field
      54E0     109          ANL     A, #ID2_0_MASK      ; use ID.2 ... ID.0 only
      C4       110          SWAP    A
      03       111          RR      A                   ; A = 4*ID.2 + 2*ID.1 + ID.0
               112          ; this value is used as an index for an array of 8 bytes
               113          ; containing the destination-addresses for the 8 different
               114          ; messages. Note, that the #RX_ARRAY_OFFSET is due to the
00A2           115          ; program counter-relative access to the array.
      2415     116          ADD     A, #RX_ARRAY_START – RX_ARRAY_OFFSET
      83       117          MOVC    A, @A + PC
               118          RX_ARRAY_OFFSET:
               119
00A5           120          ; if a message passes the acceptance-filter of the CAN
00A7           121          ; Controller, but the CPU doesn't need it, the array
               122          ; entry's value may be set to zero indicating this.
               123          ; The following <jz> instruction cares for this.
      6007     124          JZ      CAN_RX_READY
00A9           125
00AB           126          ; now copy the Data-Field (only) from CAN- to CPU memory
00AD           127          ; with the aid of the DMA-logic. Note, that a TX-DMA is
               128          ; performed when writing 8AH (DMA + address 10) into CANADR
               129          ; and a RX-DMA is performed when writing 94H (DMA + address 20)
               130          ; ... 9DH (DMA + address 29) into CANADR. Here address 22 is
               131          ; used to copy just the Data-Field.
      F5D8     132          MOV     CANSTA, A           ; data-memory address
      75DB96   133          MOV     CANADR, #CAN_RX_DMA ; starts RX-DMA at address 22
               134
00AE           135          ; the DMA-transfer is done in at maximum 2 instruction cycles.
00AF           136          ; During the transfer, neither the data-memory (RAM) nor one
               137          ; of the SFRs CANADR, CANDAT, CANCON and
00B0           138          ; CANSTA may be accessed by the CPU.
               139          ; For simplicity, two NOPs are used here.
      00       140          NOP
      00       141          NOP
00A0           142
```