502 080 01

DRAFT April 1995

P-NET

CHIP

Manual

GB *April 1995*

Proces-Data Silkeborg ApS, Navervej 10, DK-8600 Silkeborg, Denmark, phone +45-86814033, fax +45-86814088

Contents

1 Introduction

The P-NET Chip (PNC) is a small integrated circuit with two build-in asynchronous UARTs. Through FIFO'es and additional registers the PNC is able to communicate with a computer system via a parallel address/data-bus.

The PNC is a P-NET DUART (Double UART), which takes care of the lower end, time critical parts of the P-NET communication. This includes all bus access, multimaster counters, synchronization etc, in short most of the layer 2 tasks, as described in the P-NET standard. The task's left for the Host are less time critical, more complicated functions, like handling the softwire table, performing address conversion etc.

The 2 UART's in the PNC are named Port_1 and Port_2.

Port_1 can be configured to support **one** of the following modes:

0 P-NET access to ModePort1.
P-NET access to PNC parallel port.
This mode is the default mode, selected after RESET.

1 Full P-NET multimaster protocol, with baud rate 9600 or 76800 - electrical standard RS485 or IS16.
P-NET access to ModePort1.

2 DataMode, with baud rate 1200, 2400, 4800, 9600, 19200, 38400 or 76800 - electrical standard RS485 or IS16.
P-NET access to ModePort1.

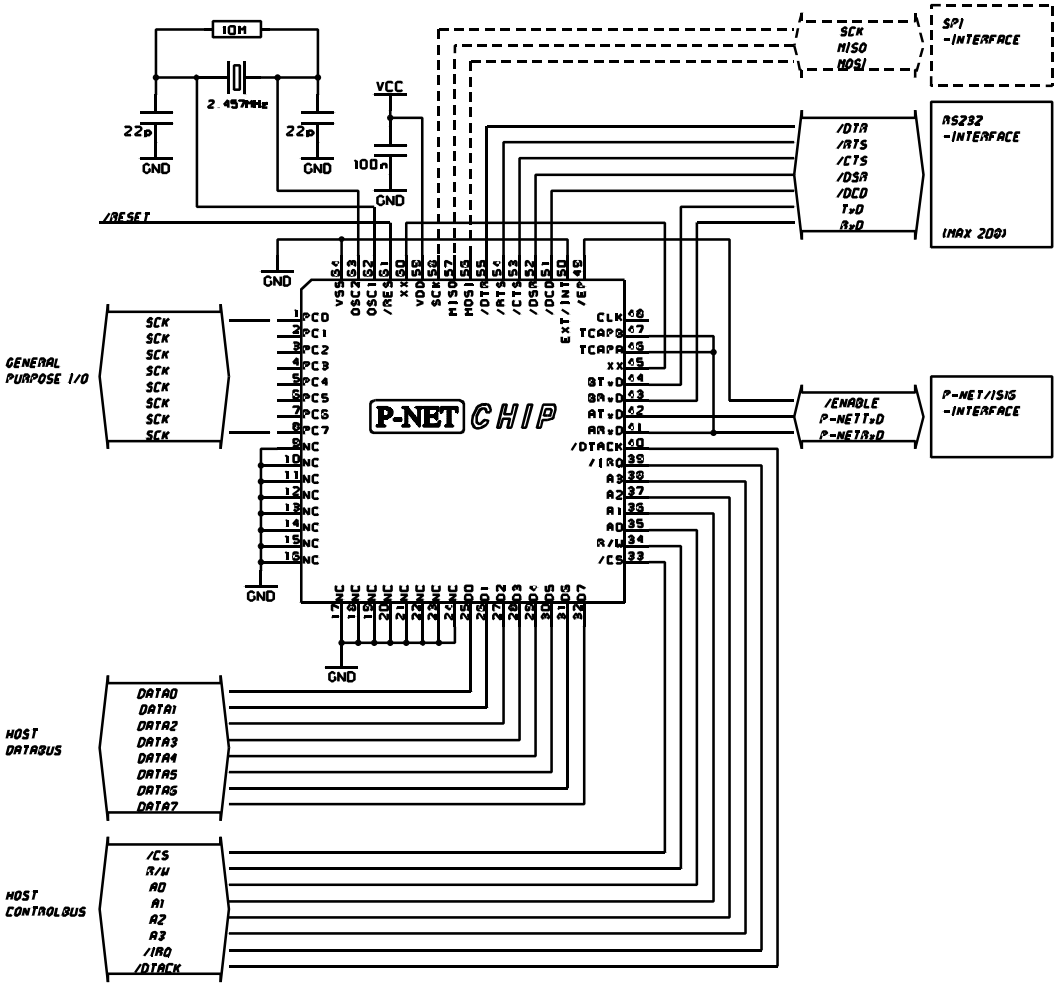
Port_2 can be configured to support **one** of the following modes:

0 P-NET access to ModePort2.
P-NET access to RealTimeClock or serial EEPROM.
P-NET access to PNC parallel port.
This mode is the default mode, selected after RESET.

1 P-NET protocol with 1 or 2 masters only, with baud rate 1200, 2400, 4800 or 9600 - electrical standard RS232.
P-NET access to ModePort2.

2 DataMode, with baud rate 1200, 2400, 4800 or 9600 - electrical standard RS232.
P-NET access to ModePort2.

Connection diagram



All not used pins on Port C should be connected to ground.

2 Data exchange betweeen Host and PNC

The exchange of data between the PNC and the Host system takes place through FIFO'es located in the PNC. The PNC contains 4 FIFO's for transferring data from Host to PNC (FIFO_Out_A/B/C/D), and 4 FIFO's for transferring data from PNC to Host (FIFO_In_A/B/C/D). Each FIFO can hold 4 bytes.

FIFO_Out_A and C are used to transfer master request's from Host to PNC Port_1 and 2 respectively - these FIFO'es are named Master_Request_FIFO_A and C.

FIFO_In_A and C are used to transfer master responses from PNC Port_1 and 2 respectively to Host - these FIFO'es are named Master_Response_FIFO_A and C.

FIFO_In_B and D are used to transfer slave request's from PNC Port_1 and 2 respectively to Host - these FIFO'es are named Slave_Request_FIFO_B and D.

FIFO_Out_B and D are used to transfer slave responses from Host to PNC Port_1 and 2 respectively - these FIFO'es are named Slave_Response_FIFO_B and D.

If a port is in DataMode, the Slave_Response_FIFO of that port is used for transmit data, and the Slave_Request_FIFO is used for receive data.

The master FIFO'es for a port in DataMode still work in P-NET mode, and can be used to access Mode for that port.

All communication between PNC and Host use the P-NET standard, except DataMode communication.

Access to registers inside the PNC (ModePort1, ModePort2, Parallel port) or SPI access (RealTimeClock, EEPROM) also use the P-NET standard, but when the request is transferred from Host to PNC, the NodeAddress consists of only 1 source address (no Destination addresses). The SPI access may take long time, so the response time may be longer than specified in the P-NET standard. The PNC does not answer with "Wait" or "Answer comes later". Instead, it performs the specified operation (maybe load in the RealTimeClock - which will take about 0.3 seconds) and then returns the answer to the Host.

When a port is used for P-NET communication, data betweeen PNC and Host are transferred in packets. The packet format is defined in the P-NET standard.

Data from Host to PNC are transferred in 4 byte blocks. When the PNC has read the 4 bytes, and is ready to receive more, it "asks for more" from the Host by inserting a certain code (<$90> <$90> <$90> <$90> for SlaveResponse, <$91> <$91> <$91> <$91> for MasterRequest and <$92> <$92> <$92> <$92> for DataMode. The "Send more" code is transferred in the corresponding FIFO (FIFO_In_A for MasterRequest etc.). FIFO_in full gives an interrupt to the Host.

Data from PNC to Host are also transferred in 4 byte blocks. The Host will thus get an interrupt each time 4 bytes are transferred from PNC to Host.

3 Master

The PNC is able to perform most of the P-NET layer 2 tasks, like bus access, synchronization etc., needed for the P-NET multimaster system (Port_2 does not support the multimaster system).

When the Host wants to initiate a P-NET transmission, it inserts a Master_Request_Packet in the Master_Buffer for that port. When the Master_Request_FIFO is available (that is: the former Master_Request - if any - completely transferred to the PNC, and the response completely transferred to the Host), the Host starts transferring the packet from the Master Buffer to the PNC through the Master_Request_FIFO.

The Master_Request_Packet is initiated with a

Master_Request_Header: <$01> when layer 2 shall use reduced error checking (only 1 error detect byte)

or with a

Master_Request_Header: <$02> when layer 2 shall use normal error checking (2 error detect bytes)

The header is followed by the

Node_Address_Field: <length> <dest.addr.> <dest.addr> ... <source addr.> <source addr.>..

Only the actual number of node addresses, defined in the first byte, <length>, are sent. If the transmission is for SPI access or registers inside the PNC, the Node_Address_Field holds no dest.addr. and only 1 source addr.

The node address is followed by the

Control_Status: <Control_Status>

Refer to the P-NET standard for definition of Control_Status.

Control_Status is followed by the

Info_Length: <info_Length>

Refer to the P-NET standard for definition of Info_Length.

Info_Length is followed by the

Info_Field: <Info_1> <Info_2>...

Only the actual number of Info bytes, defined in the Info_Length, are sent. However, the Host

always transfers 4 byte at a time. Thus, the last transfer will hold between 0 and 3 dummy byte.

During transfer of the Master_Request_Packet the PNC sends the "Send more" code (4 times <$91>) in the corresponding Master_Response_FIFO each time it has emptied the Master_Request_FIFO and expects more data. However, if the code <$80> <$80> <$80> <$xx> is sent by the PNC, it means that an error has occured. The Host must then immidiately stop transferring the Master_Request_Packet, and report the error. The following errors are possible:

<$FE> The Host transferred data to slowly to the Master_Request_FIFO, which has caused the PNC to immidiately stop transmitting to layer 1. The error is called HostTooSlow.

<$90> Net shortcircuit

<$98> Port not master

<$A0> Out of sync

The errorcodes (except HostTooSlow) correspond to Control/Status, as defined in the P-NET standard. HostTooSlow must be converted (in the Host) into No Response, before the error is reported to the user. The origin of the error is set to be the port.

During transfer of the first 8 bytes in the Master_Request, it is possible that the PNC starts to receive a Slave_Request from layer 1. The PNC transfers the Request to the Host through the corresponding Slave_Request_FIFO, and the Host returns the Response through the Slave_Response_FIFO. If an error occurs during the transfer of the Slave_Response, the PNC will send the code <$82> <$82> <$82> <$xx> through the Master_Response_FIFO for the same port. So, during transfer of the first 8 bytes of a Master_Request, the Host must look for the following 3 codes in the corresponding Master_Response_FIFO:

<$91>: Send next 4 bytes in Master_Request
<$80>: Stop Master_Request transfer - read ErrorCode in Master_Response_FIFO
<$82>: Stop Slave_Response transfer on this port - read ErrorCode in Master_Response_FIFO - then continue Master_Request transfer

When the PNC is ready, and finds out, that a request from the Host is pending, it reads the first 4 bytes from the packet, stores them in an internal buffer, asks for more (transfers "Send more" code), and waits until the Master-Request_FIFO is full again before it is ready to start sending.

However, if one of the errors "Port not master" or "Out of sync" are present, the PNC reads only 4 bytes from the Master_Request_FIFO and transfers the corresponding errorcode in the Master_Response_FIFO.

If none of the above mentioned errors are present, the PNC starts sending a frame when it is allowed to do so, and the Master_Request_FIFO is full again (when the HOST has sent 8 byte), according to the P-NET multimaster rules. The node address is not sent directly as it was received from the Host, but is converted to a simple, extended or complex node address field, depending on the length of the field. Each time the PNC has sent 4 bytes to layer 1 from it's internal buffer,

it will read the next 4 bytes from the Master_Request_FIFO into it's internal buffer. If they are not present, the PNC immidiately stops sending data to layer 1, and transfers the corresponding errorcode to the Master_Response_FIFO.

During transmission to layer 1, the PNC generates the Error_Detect_Code(s). After the Info_Field the Error_Detect_Code(s) is (are) sent.

When the PNC has sent the frame, it is waiting for a response, except if the first destination address = 126 or the last source address = nil ($80). In this case the PNC transfers the code <$81> <$81> <$81> <$FF> to the Host, in the Slave_Request_FIFO. If this is not the case, 5 situations are possible:

1: Receive a response within 38 bit periods, with "Answer comes later"

The received frame is transferred to the Host, as normally. The Host inserts the original packet in a WaitQueue, waiting for the "later answer".

2: Receive a response within 38 bit periods, with "Busy" or "Wait".

The received frame is transferred to the Host, as normally. The Host inserts the original packet in the Master Buffer again - if not time out.

3: Receive a normal response within 38 bit periods:

The received frame is transferred to the Host.

4: No response within 38 bit periods:

Errorcode <$81> <$81> <$81> <$00> is transferred to the Host in the Slave_Request_FIFO.

5: An error occured during response reception:

If an error - like framing error, instruction error, error detect failure etc. - occurs, or if the slave stops transmitting the response, the PNC immidiately stops receiving from layer 1, and transmitting to the Host through the Master_Response_FIFO. Instead, it sends the errorcode <$81> <$81> <$81> <$xx> to the Host through the Slave_Request_FIFO for the same port. The errorcode corresponds to the errortype, as described in the P-NET standard. If the slave stopped transmitting, the errorcode is "NO ANSWER" as if it had not started at all.

The Master_Response_Packet is transferred from PNC to Host in the following steps:

When the PNC has received Control/Status and the next 3 bytes (if the response is that long) from layer 1, these 4 bytes are transferred to the Host. This goes on, by sending 4 bytes at a time, until the whole response is sent. The node address field is not transferred.

The last transfer may consist of less than 4 info bytes. The PNC fills up with dummy'es, <$xx>. The last 4 byte transfer from PNC to Host will thus hold between 0 and 3 dummy bytes.

4 Slave

In general, the PNC receives a request frame for this node, converts it to a Slave_Request_Packet and transfers the packet to the Host through a Slave_Request_FIFO. The Host interpretes and services the request, and then returns a Slave_Response_Packet to the PNC, through a Slave_Response_FIFO.

However, it is not possible to let the PNC receive the whole frame before it starts to convert it to a packet and transfer it to the Host, because the Host would not be able to interprete and service the request fast enough. Therefore, the transfer of the Slave_Request_Packet is done 4 bytes at a time. At the end, the PNC fills up with dummy byte, <$xx>, so that an even number of 4 byte transfers are made. The last transfer from PNC to Host will thus hold between 0 and 3 dummy byte.

Slave_Request_Header: <$04>

The header is followed by the

Node_Address_Field: <length> <dest.addr.> <dest.addr> ... <source addr> <source addr>..

and the rest of the P-NET frame, as it is received from layer 1. The error detect codes are not transferred to the Host.

If an error occured during reception of the frame from layer 1, like framing error or error detect failure, or if the master (another node) stopped transmitting the request, the PNC immidiately stops receiving from layer 1, and transmitting to the Host through the Slave_Request_FIFO. Instead, the error code <$82> <$82> <$82> <$xx> is sent to the Host through the Master_Response_FIFO, where <$xx> corresponds to the error, as described in the P-NET standard.

If the Host received a complete and correct Slave_Request_Packet from the PNC, it must start to return a Slave_Response_Packet to the PNC as soon as possible, and not later than defined in chapter "Speed demands on Host". The PNC starts transmitting to layer 1 as soon as the Host started to return the Slave_Response_Packet, though not before the line has been idle for at least 11 bit periods. If the first destination address in the request was = 126, or the last source address was = nil, the Host shall not return anything. The PNC expects no answer from the Host.

The Slave_Response_Packet is transferred from Host to PNC like this:

> Within 3 byte's time from Slave_Request_FIFO full (last part of Slave_Request):
>
> Node_Address_Field: <dest.addr.> <source addr.>

The Node address in the response packet always consists of only 2 bytes, therefore no length is transferred.

Within 1 byte's time later:

Control/Status, Info_Length

Within 2 byte's time after "send more" interrupt (Slave_Request_FIFO = <$90> <$90> <$90> <$90>:

Info_Field - 4 bytes

and so on, till the complete info field is sent. Alse the last transfer will consist of 4 bytes, and will thus hold between 0 and 3 dummy'es (<$xx>).

If an error occurs during transmission of the Slave_Response_Frame (HostTooSlow) the PNC sends the errorcode <$83> <$83> <$83> <$FE> to the Host through the Slave_Request_FIFO.

5 Speed demands on Host

Interrupt to the Host is generated on FIFO_In full. This means, that the number of interrupts (and thus the total time consumption) is low, as the Host will only get an interrupt for each 4 bytes received from the PNC.

This gives the following speed demand on the Host:

During master request transfer to PNC:

Host must be able to write 4 bytes to the Master_Request_FIFO within 4 byte's time after the interrupt for Master_Response_FIFO full was asserted.

During master response transfer from PNC:

Host must be able to read 4 bytes from the Master_Response_FIFO within 4 byte's time after the interrupt for Master_Response_FIFO full was asserted.

During slave request transfer from PNC:

Host must be able to read 4 bytes from the Slave_Request_FIFO within 2 byte's time after the interrupt for Slave_Request_FIFO full was asserted. When the last part of the request is received, the Host must be able to write the node address to the Slave_Response_FIFO (always 2 bytes) within 3 byte's time after the interrupt for Slave_Request_FIFO full was asserted, and the next 2 bytes (Control/Status and InfoLength) within 1 byte's time later.

During slave response transfer to PNC:

After having sent the node address, the Host must be able to send Control/Status and Info length within 1 byte's time after the Slave_Response_FIFO became empty.

When this is done, the Host must be able to send 4 bytes to the Slave_Response_FIFO within 2 byte's time after the interrupt for FIFO empty was asserted.

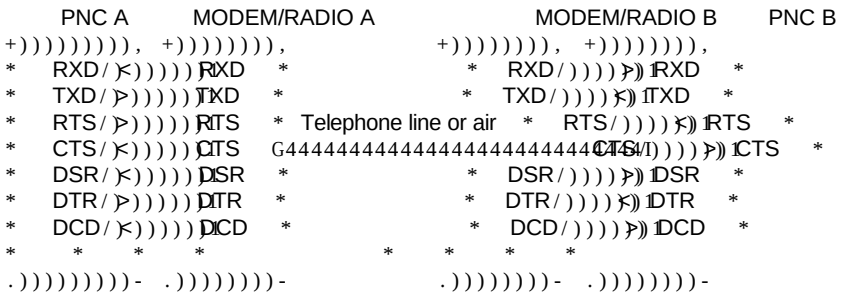
So, in general, if the PNC is used for master transmissions only, the Host must be able to respond to interrupt within 4 byte's time, and if the PNC is used for slave transmissions the Host must be able to respond within 2 byte's time (due to the demands in the P-NET standard on very fast response from slaves).

6 P-NET communication on RS232 line

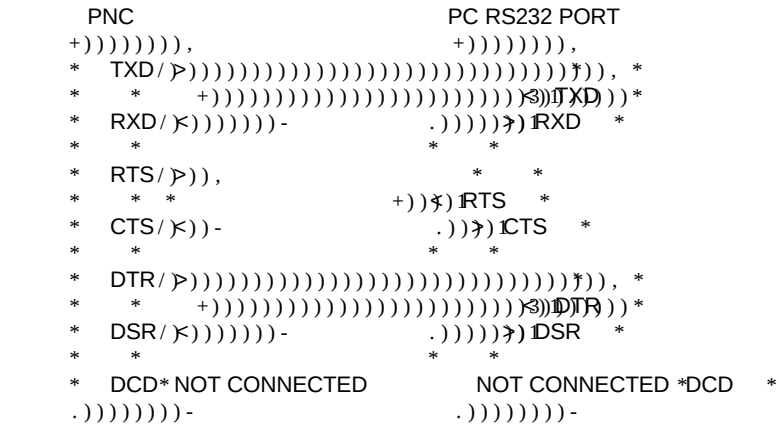
When the electrical standard RS232 is used, no standard P-NET multimaster synchronization is performed, but it is still possible to have 2 masters.

The port is intended for modem or radio communication etc., or for communication with a PC with an RS232 port.

In case of modem or radio communication, the system is connected this way:



In case of communication between PNC and PC with RS232 port, the system is connected this way:



When a node wants to send a master request, it is going through the following sequence:

1: Wait here till DSR activated. (max 5 second - if still not activated report error to host).
2: Deactivate DTR.
3: Wait 5 msec.
4: Check DSR - if still activated go on to 5 - otherwise activate DTR and go to 1.
5: Activate RTS.
6: Wait here till CTS activated. (max 1 second - if still not activated deactivate RTS, activate DTR and report error to host).
7: Send request.
8: Deactivate RTS.
9: Activate DTR.

Now wait max 1 second for the response from the slave.

When a node wants to send a slave response, it is going through the following sequence:

1: Activate RTS.
2: Wait here till CTS activated. (max 1 second - if still not activated deactivate RTS and forget all about it).
3: Send response.
4: Deactivate RTS.

This system gives a kind of multimaster access to the line - the master node who comes first is allowed to send. When a master node is waiting for response from the slave, it is not allowed to send a new master request.

When the line is idle, both modules holds DTR activated, signaling "I am ready to receive a request".

DTR active after InitPort.

7 Datamode

When a port is in Datamode, data from PNC to Host is transferred in the Slave_Request_FIFO for that port, and data from Host to PNC is transferred in the Slave_Response_FIFO.

In principle, the PNC is transparent, meaning that data from the net is transferred directly to Host, and data from Host are transferred directly to the net.

Datamode IN

Data from the net are transferred from PNC to Host in 4 byte blocks. The first byte denotes the number of data-bytes in the block, and can hold the values 1, 2 or 3 for 1, 2 or 3 byte valid data respectively. If the value of this byte is less than 3, the FIFO is filled up with dummy'es.

The PNC activates DTR if in RS232 auto mode, and receives data from the net. When 3 byte are received, or the line goes Idle (consecutive "1" on the net for 10 bit-times, a datablock is transferred to the Host through the Slave_Request_FIFO.
The RS232 handshake input/outputs is not used if the PNC is in Manual mode (Mode-PortX.Manual:= TRUE)

Datamode_In_Header: <1> or <2> or <3>

Data: 1, 2 or 3 byte data

Dummy: 0, 1 or 2 dummy byte

If the Slave_Request_FIFO is not empty when the PNC is ready to transfer data to the Host, the PNC de-activates DTR if in RS232 Auto mode, signaling the transmitter to stop sending. When the Slave_Request_FIFO becomes empty, the PNC transfers the datablock, and activates DTR again.

InitPort activates DTR.

Datamode OUT

Data from Host to the net are transferred to the PNC in 4 byte blocks. When the Host has data to be sent, it sends a datablock with a header denoting the number of valid data byte, followed by data. If there is less than 3 data byte to be sent, the datablock is filled up with dummy'es.
When the PNC has read the block, and is ready to receive more, it "asks for more" from the Host by inserting a send more code (<$92><$92><$92><$92>) in the Slave_Request_FIFO.

Datamode_Out_Header: <1> or <2> or <3> or <$81> or <$82> or <$83>

Bit 7 = 1 indicates that it's the last block, so the PNC must stop sending "send more" codes and deactivate RTS if in RS232 auto mode.

Data: 1, 2 or 3 byte data

Dummy: 0, 1 or 2 dummy byte

The Host must not send data to the PNC until the Slave_Response_FIFO is empty.

If the PNC is in RS232 Manual Mode (Port 2) the handshake signals are not used.

If the PNC is in RS485 or IS16 mode (Port1) the output "EP" is activated when a block is ready to be sent, and deactivated when the complete block is sent - that is when the host has sent the Datamode_Out_Header $81, $82 or $83.

If the PNC is in RS232 auto mode (Port 2), the handshake signals are used as follows:

When the first block is received from the Host, the PNC goes through the following sequence:

1: Check DSR - if activated go on to 2.
2: Activate RTS.
3: Wait here till CTS activated.

For each byte to send, the PNC goes through the following sequence:

1: Wait here till DSR activated
2: Send data.

When the last byte is sent (the host has sent the Datamode_Out_Header $81, $82 or $83, and data from that block are sent) the PNC deactivates RTS.

This systems gives the following feature: If DSR is not activated (because the receiver has deactivated DTR), the PNC stops transmitting, and stops reading from the Slave_Response_FIFO, which will make the Host stop transferring data.

Connect nodes as described in section "P-NET communication on RS232 line".

8 Softwire table in PNC

The P-NET Chip itself holds the following Softwire table:

2 SW entries for defining node adress, number of masters, protocol etc. for the 2 communication ports - ModePort1 and ModePort2.

2 SW entries for an 8 bit general purpose digital I/O port, Port B.

1 SW entry for an 8 bit I/O port for RS232 handshake signals.

2 SW entries for SPI system - refer to SPI system chapter.

The communication port SW entries and the general purpose I/O port are accessed using the P-NET protocol, simply by accessing them as any normal P-NET transmission, except no Destination Address is specified in the node address field. This means, that when the packet from the Host is received in the PNC, the PNC itself will act as if it was the slave connected to the P-NET, interprete and service the request, and return a response to the Host. Now, the Host will act as if a real P-NET transmission had been performed.

The Softwire table inside the PNC lokks like the following:

SWNo	Data	Type
01	ModePort1	RECORD
02	ModePort2	RECORD
03	DigIO Port_C Data	BYTE
04	DigIO Port_C Data_Direction	BYTE
05	Handshake port	BYTE
06	RTC	RECORD
07	EEPROM	ARRAY

ModePort1 and ModePort2:

These variables are of the following type:

```
Record
      PNETNo          : BYTE;
      NoOfMasters     : BYTE;
      Protocol        : ProtocolType;
      ElStd       : ElStdType;
      Baudrate        : LONGINTEGER;
      Sumcheck        : BOOLEAN;  (* 0: 1 byte, 1: 2 byte *)
      Parity          : ParityType;
      Manual          : BOOLEAN;  (* 0: Auto, 1: Manual *)
End
```

PNETNo defines the modules P-NET nodenumber on the port in question.

NoOfMasters defines the number of masters on the port in question.

Protocol can hold the following values:
Disabled, PNETmode, DatamodeIn, DatamodeOut, DatamodeInOut

The following functions are included, depending on the value in Protocol:

- Disabled:
  P-NET access to RealTimeClock or serial EEPROM. (only port 2)
  P-NET access to PNC parallel port.
  P-NET access to ModePortx. (mode for this port)
  - this is the default mode, selected after RESET of the PNC.

- PNETmode:
  P-NET mode.
  P-NET access to PNC parallel port.
  P-NET access to ModePortx.

- DatamodeIn:
  Datamode In.
  P-NET access to PNC parallel port.
  P-NET access to ModePortx.

- DatamodeOut:
  Datamode Out.
  P-NET access to PNC parallel port.
  P-NET access to ModePortx.

- DatamodeInOut:
  Datamode In + Out.
  P-NET access to PNC parallel port.
  P-NET access to ModePortx.

ElStd can hold the following values:
0: RS485 for Port 1
and the following values:
2: RS232 for Port 2.

Baudrate can hold the following values:
76800, 38400, 19200, 9600, 4800, 2400, 1200 for Port 1 (Datamode)
76800 or 9600 for Port 1 (P-NET)
and the following values:
9600, 4800, 2400, 1200 for Port 2.

Parity can hold the following values:
None, Address-data

Manual defines, whether handshake signals for RS232 in datamode are controlled automatically, as described in chapter Datamode. If Manual is TRUE, handshake signals are not controlled by the PNC, but must be controlled from the host using P-NET access to PNC parallel port.

ModePort may be changed from the Host. Whenever ModePort is transferred to the PNC, the complete variable should be transferred. After transferring ModePort, the PNC initializes internal registers according to the selected Mode for the Port in question.

DigIO Port_C Data:

This read/write 8 bit register holds the input/output data for Port C.

After reset, all bits in this register are cleared, so when any bit is configured to output, the output is cleared.

DigIO Port_C Data_Direction:

This read/write 8 bit register holds the data direction for Port C. Writing a "1" to any bit configures the corresponding bit in the port b data register as an output; conversely, writing any bit to "0" configures the corresponding port bit as an input.

After reset, all bits are cleared, meaning all port bits are configured as input.

Handshake port:

This read/write 8 bit register holds the input/output data for the Handshake port.

Bit 0: DCD (Data Carrier Detect) Input
Bit 1: DSR (Data Set Ready) Input
Bit 2: CTS (Clear To Send) Input
Bit 3: RTS (Request To Send) Output
Bit 4: DTR (Data Terminal Ready) Output
Bit 5-7: Not connected

9 SPI system

The SPI system is used for accessing integrated circuits, which are connected to the PNC through the SPI interface lines.

The SPI system is a synchronous, serial LINK, where the build-in facilities in the PNC operates on CLK (clocksignal), MOSI (MasterOutSlaveIn) and MISO (MasterInSlaveOut). These three signals are connected from the PNC to the integrated circuits (EEPROM and/or RealTimeClock).

The serial EEPROM must be of type NMC93CS56/CS66 or similar. The RealTimeClock (RTC) must be of type S3520 (Seiko).

ChipSelect for EEPROM is connected to PNC Port C, bit 0. ChipSelect for RTC is connected to PNC Port C, bit 1. Write for RTC is connected to PNC Port C, bit 2.

SPI communication is performed as a P-NET communication, with no Destination addresses (in the same way as access of Modeport). The Port used (only port 2) must be disabled (Protocol = Disabled).

2 softwire numbers are used for SPI access:

SWNo	Data	Type
06	RTC	RECORD
07	EEPROM	ARRAY

EEPROM:

The serial EEPROM is seen as an array of WORD. The size of the array depends on the type of EEPROM selected.

Only LOAD and STORE operation can be performed (AND, OR, TAS, LONG LOAD and LONG STORE are illegal). Only 1 element (2 byte) can be loaded or stored in the EEPROM pr. transmission.

The address in the EEPROM is selected be means of the offset in the P-NET request.

RTC

The RealTimeClock is a RECORD of the following type:

```
Record
    Second        : BYTE;      (0 - 59)
    Minute        : BYTE;      (0 - 59)
    Hour          : BYTE;      (0 - 23)
    Day           : BYTE;      (1 - 7, sunday = 7)
    Date          : BYTE;      (1 - 28/29/30/31)
    Month         : BYTE;      (1 - 12, january = 1)
    Year          : BYTE;      (0 - 99, modulus 100 of the year)
    MasterReset   : BOOLEAN;   (TRUE: MasterReset)
End
```

The MasterReset flag is automatically set when power is applied to the RTC. The RTC must be battery powered.

The MasterReset flag is reset be writing to the RTC.

RTC accesses must always access the complete variable (8 byte).

A read access may take up to 300 ms. A write access may take up to 1 second.

10 Register reference

This chapter gives a register overview and describes technical details on how the PNC is initialized etc.

Register overview:

Reg.No	Reg.Name
$00	FIFO_Out_A
$01	FIFO_Out_B
$02	FIFO_Out_C
$03	FIFO_Out_D
$04	FIFO_In_A
$05	FIFO_In_B
$06	FIFO_In_C
$07	FIFO_In_D
$08	FIFO_A_Status
$09	FIFO_B_Status
$0A	FIFO_C_Status
$0B	FIFO_D_Status
$0C	Interrupt_Status
$0D	Interrupt_Mask
$0E	Mode
$0F	Un_Used

After a hardware RESET, the PNC must be initialized according to the following:

1: Wait min. 2 msec.
2: Clear all FIFO'es by writing $4F to the "Mode" register.
3: Select interrupt on FIFO Full by writing $0F to the "Mode" register
4: Enable interrupt by writing $AA to the "Interrupt_Mask" register.
5: Wait min. 2 msec.

When an interrupt occurs, the Interrupt_Status register may be examined, to check which FIFO caused the interrupt. The Interrupt_Status register holds the following:

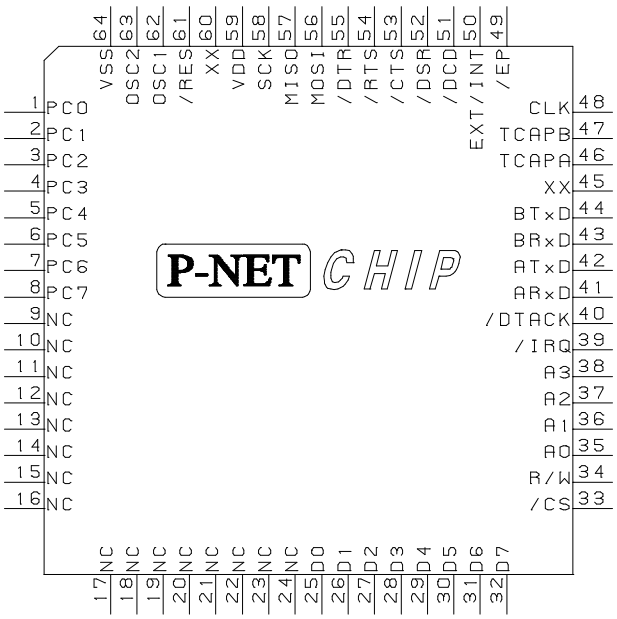
Bit[1]: FIFO_In_A full
Bit[3]: FIFO_In_B full
Bit[5]: FIFO_In_C full
Bit[7]: FIFO_In_D full

The 4 FIFO status registers holds the following (for the 4 FIFO'es respectivily):

Bit[0]: TEMTY - Transmitter Empty
Set on reset or when FIFO_Out is empty.
Clear when FIFO_Out is not empty.
Bit[1]: TFULL - Transmitter Full
Set when FIFO_Out is full.
Clear when FIFO_Out is not full.
Bit[2]: TRDY - Transmitter Ready
Set when a byte is waiting in FIFO_Out to be read by the PNC.
Clear when no byte is waiting - FIFO_out is empty.
Bit[3]: TOR - Transmitter Overrun
Set when the Host sends a byte to FIFO_Out as it was already full.
Cleared on reset or when the Host reads the status register.
Bit[4]: REMTY - Receiver Empty
Set when FIFO_In holds no data for the Host.
Clear when FIFO_In is not empty.
Bit[5]: RFULL - Receiver Full
Set when FIFO_In holds 4 byte for the Host.
Clear when FIFO_In is not full.
Bit[6]: RRDY - Receiver Ready
Set when FIFO_In holds at least one byte for the Host.
Clear when FIFO_In is empty.
Bit[7]: ROR - Receiver Overrun
Set when the PNC sends a byte to FIFO_In as it was already full.
Cleared on reset or when the PNC reads the status register.

11 Functional PIN description

All signal inputs on the P-NET CHIP except /RES and OSC1 are TTL compatible.



VDD and VSS

Power is supplied to the P-NET CHIP via these two pins. VDD is the positive supply and VSS is ground.

It is in the nature of CMOS design that very fast signal transitions occur on the P-NET CHIP pins. These short rise and fall times place very high short-duration current demands on the power supply.
To prevent noise problems, special care must be taken to provide good power supply by-passing at the P-NET CHIP. By-pass capacitor should have good high-frequency characteristics and be as close to the P-NET CHIP as possible.

/RES

This active low input is used to reset the P-NET CHIP. Applying a logic zero to this pin forces the P-NET CHIP to a known start-up state.

OSC1/OSC2

These pins provide control input for an on-chip clock oscillator circuit. A crystal connected to these pins provides the oscillator clock.

Crystal Parameters

Parameter	Value	Unit
Frequency	2.457	MHz
$RS_{(max)}$	400	Ohm
C0	5	pF
C1	0.008	uF
Q	30k	-

I/O pins

RS485/IS16 interface

ATxD (Transmit Data, Port A)
Serial-data output to the P-NET.

ARxD (Receive Data, Port A)
Serial-data input from the P-NET.

/EP (Enable P-NET, Port A)
This active low output is used to Enable the P-NET interface. (/Drive enable)

RS232 interface

BTxD (Transmit Data, Port B)
RS232 Serial-data output.

BRxD (Receive Data, Port B)
RS232 Serial-data input.

/DTR (Data Terminal Ready, Port B)
RS232 Data Terminal Ready output signal.

/RTS (Request to Send, Port B)
RS232 Request To Send output signal.

/CTS (Clear To send, Port B)
RS232 Clear To Send input signal.

/DSR (Data Set Ready, Port B)
RS232 Data Set Ready input signal.

/DCD (Data Carrier Detect, Port B)
RS232 Data Carrier Detect input signal.

PORT C

Port C consist of eight bi-directional pins. The direction and state of each pin is software programmable. All pins are configured as input during power-on reset.

SPI facility pins

The SPI (Serial Peripheral Interface) facility offers an easy to use connection to for example EEPROMs and RealTimeClocks.

SCK

The SPI SerialClocK output signal.

MISO

The SPI MasterInSlaveOut input signal.

MOSI (Master Out Slave In)

The SPI MasterOutSlaveIn output signal.

Host interface pins

/DTACK (Data Transfer ACKnowledge)

The tri-state active-low open-drain Data Transfer Acknowledge output signal is asserted during read and write cycles to indicate the proper transfer of data between the Host and the P-NET CHIP.

The total Host bus access time is the time from /CS asserted to /DTACK asserted. Since the Host bus accesses are asynchronous with respect to the P-NET CHIP it is necessary to provide a maximum delay from /CS asserted to /DTACK asserted. To achieve this an on-chip clock delay is incorporated.

Note: For a write cycle, if /CS is negated within one clock cycle of it being recognised (i.e. the first rising edge of the clock), then /DTACK may not be generated. In this case data will be latched on the negating edge of /CS.

/IRQ (InterruptReQuest)

This active-low open-drain output pin signals to the Host that the P-NET CHIP have data in it's buffer.

When the /IRQ pin is asserted it indicates that an interrupt condition has occurred. This pin remains asserted until the Host responds to the interrupt and read the data register.

A0-A3

The register select lines select the P-NET CHIP internal registers during read/write operations.

D0-D7 (Data 0 - Data 7)

These bi-directional tri-state data lines are used to transfer commands, data and status informations between the Host and the P-NET CHIP. D0 is the least significant bit.

R/W (Read / Write)

The READ/WRITE input pin, R/W is driven high by the Host when it reads and low when it writes. Read or Write cycles are initiated by asserting the chip select input /CS.

/CS (/ChipSelect)

The active low /CS input enables data transfers between the Host and the P-NET CHIP on data lines D0 to D7. Data transfers are controlled by Read/Write, /Chip-Select and the register select inputs A0-A3. When /CS is high the data lines D0-D7 are placed in a high impedance state.

CLK

External Clock input

EXT/INT (Option)

This input selects between external or internal clock. When using a crystal this input must be connected to VSS.

Misc. connections

XX-XX

External connection. (Connect pin 45 and pin 60)

TCAPA-TCAPB

External connection (Connect pin 41 and pin 46 and pin 47)

NC

External connection (Connect to VSS)

12 Electrical specifications

This section contains the electrical specifications and associated timing information for the P-NET CHIP.

Maximum ratings

Rating	Symbol	Value	Unit
Supply Voltage	VDD	-0.3 to +7.0	V
Input Voltage	V_{in}	VSS-0.3 to VDD+0.3	V
Operating Temperature Range	T_A	-40 to +85	°C
Storge Temperature Range	T_{stg}	-65 to +150	°C
Current Drain per pin (note 2)	I_D	25	mA

Note 1) All voltages are with respect to VSS.

Note 2) Maximum current drain per pin is for one pin at a time, limited by an external resistor. (- excluding VDD and VSS)

Note 3) This device contains circuitry designed to protect against damage due to high electrostatic voltage or electric fields. However, it is recommended that normal precautions be taken to avoid the application of any voltage higher than those given in the maximum Ratings table to this high impedance circuit. For maximum reliability all unused inputs should be tied to either VSS or VDD.

Thermal characteristics

Package Thermal Characteristics

Charateristics	Symbol	Value	Unit
Thermal Resistance	JA	50	°C/W

DC electrical characteristics

Characteristic	Symbol	Min	Typ	Max	Unit
Output Voltage I_{LOAD} = 10μA I_{LOAD} = 10μA	 V_{OH} V_{OL}	 VDD - 0.1 -	 - -	 - 0.1	 V V
Output High Voltage Port A-C (ILOAD = -0.8 mA)	V_{OH}	VDD-0.8	-	-	V
Output Low Voltage Port A-C (ILOAD = +1.6 mA)	V_{OL}	-	-	0.4	V
Input High Voltage All ports, OSC, /RES	V_{IH}	0.7VDD	-	-	V
Input Low Voltage All ports, OSC, /IRQ, /RES	V_{IL}	-	-	0.2VDD	V
Output High Voltage D0-D7 (ILOAD = -0.8 mA)	V_{OH}	VDD-0.8	-	-	V
Output Low Voltage D0-D7, OSC1, /IRQ, /DTACK (ILOAD = +1.6 mA)	V_{OL}	-	-	0.4	V
Input High Voltage D0-D7, A0-A3, /CS, R/W	V_{IH}	0.7VDD	-	-	V
Input Low Voltage D0-D7, A0-A3, /CS, R/W	V_{IL}	-	-	0.2VDD	V
High-Z Leakage Current Port A-C	I_{IL}	-	-	+/- 10	μA
Input Current /RES, OSC1	I_{IN}	-	-	+/- 1	μA
Capacitance Ports (as input or output) /RES	 C_{out} C_{in}	 - -	 - -	 12 8	 pF pF
Hysteresis /RES	V_{HYST}	-	1.0	-	V

Note 1) All measurements are taken with suitable decoupling capacitor across the power supply to suppress the transient switching currents inherent in CMOS designs.

Note 2) Typical values are at mid point of voltage range and at 25 °C only.

AC electrical characteristics

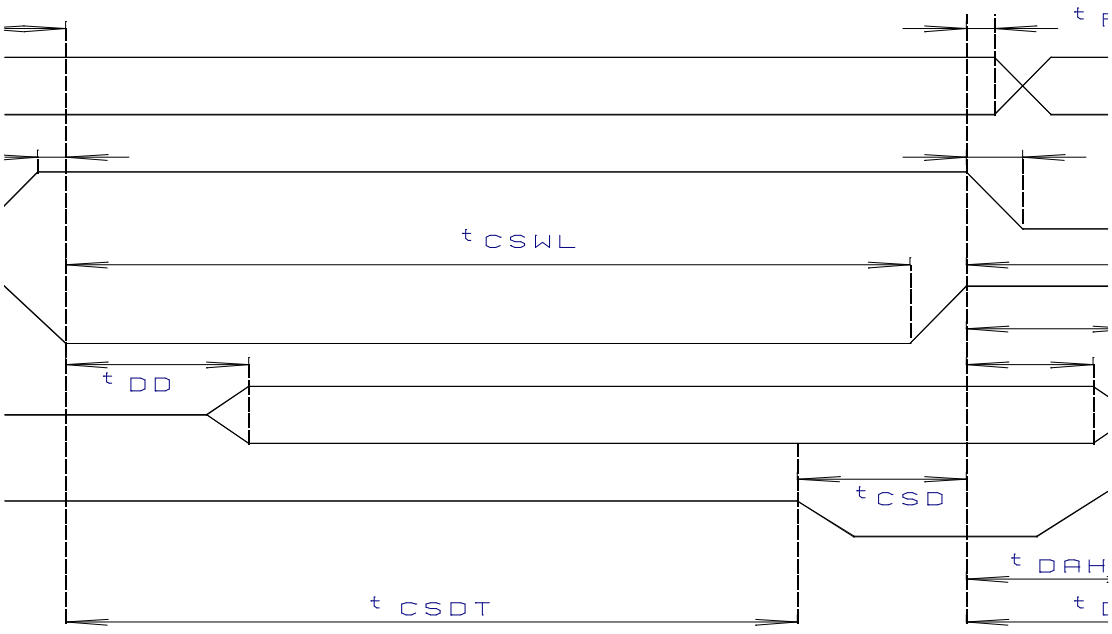
(VDD=5.0 Vdc+/- 10%, VSS=0 Vdc, TA=TL to TH)

Characteristic	Symbol	Min	Max	Unit
Frequency of Operation Crystal	f_{OSC}	2.457	2.457	MHz
Crystal Oscillator startup time	t_{OXOV}	-	100	ms
/RES pulse width	t_{RL}	2	-	μs
Power-on Reset delay	t_{PORL}	4	4	ms

HOST interface timing

Host Read Cycle Bus Timing

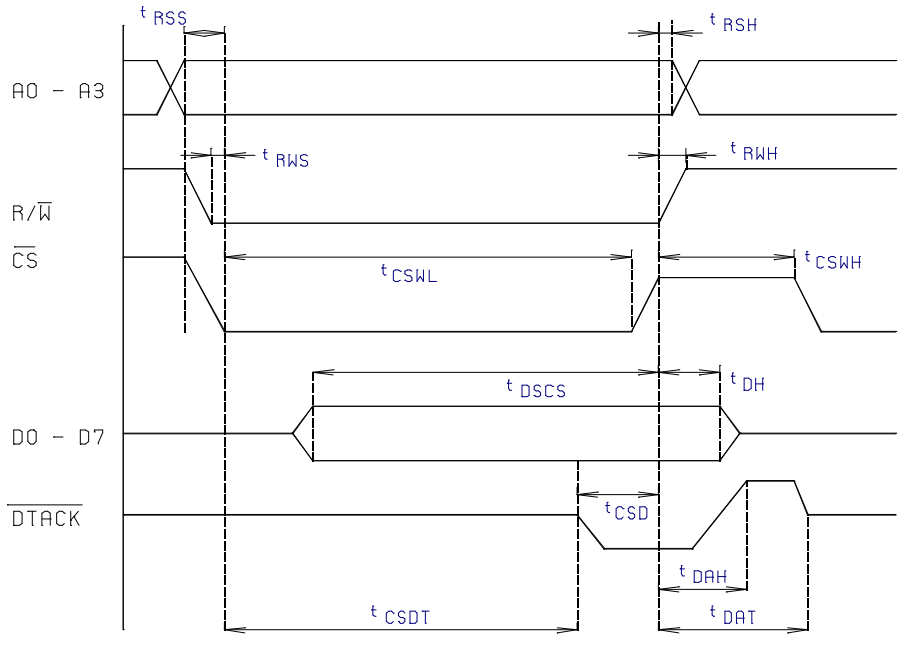
Characteristic	Symbol	Min	Max	Unit
A0-A3 Setup Time to /CS Asserted	t_{RSS}	10	-	ns
R/W Setup Time to /CS Asserted	t_{RWS}	10	-	ns
/CS Pulse Width Asserted	t_{CSWL}	130	-	ns
Data Valid from /CS Asserted	t_{DD}	-	100	ns
/CS Asserted to /DTACK Asserted	t_{CSDT}	-	2.5	μs
/CS Negated from /DTACK Accerted	t_{CSD}	20	-	ns
A0-A3 Hold Time from /CS Negated	t_{RSH}	0	-	ns
R/W Hold Time from /CS Negated	t_{RWH}	0	.	ns
Data Hold Time from /CS Negated	t_{DH}	0	-	ns
Data Bus floating from /CS Negated	t_{DF}	-	50	ns
/DTACK Negated fom /CS Negated	t_{DAH}	-	50	ns
/DTACK Hi-Z from /CS Negated	t_{DAT}	-	75	ns
/CS Negated Pulse Width	t_{CSWH}	70	-	ns



READ CYCLE BUS TIMING DIAGRAM

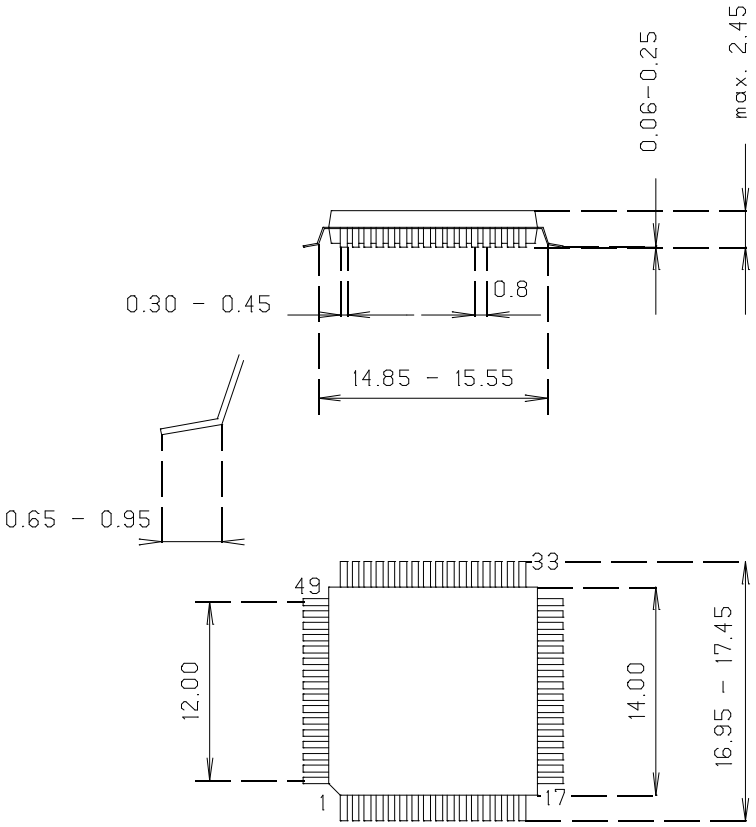
Host Write Cycle Bus Timing

Characteristic	Symbol	Min	Max	Unit
A0-A3 Setup Time to /CS Asserted	t_{RSS}	10	-	ns
R/W Setup Time to /CS Asserted	t_{RWS}	10	-	ns
/CS Pulse Width Asserted	t_{CSWL}	130	-	ns
/CS Asserted to /DTACK Asserted	t_{CSDT}	-	2.5	μs
/CS Negated from /DTACK Asserted	t_{CSD}	20	-	ns
A0-A3 Hold Time from /CS Negated	t_{RSH}	0	-	ns
R/W Hold Time from /CS Negated	t_{RWH}	0	-	ns
Data Hold Time from /CS Negated	t_{DH}	10	-	ns
Data Setup Time to /CS Negated	t_{DSCS}	-	50	ns
/DTACK Negated from /CS Negated	t_{DAH}	-	50	ns
/DTACK Hi-Z from /CS Negated	t_{DAT}	-	75	ns
/CS Negated Pulse Width	t_{CSWH}	70	-	ns



WRITE CYCLE BUS TIMING DIAGRAM

13 Mechanical specifications



QUAD FLAT PACK 64 (QFP 14x14-64)
COMPONENT DIMENSIONS