

UDP Layer

Support routines

Fri, Jul 12, 1996

For support of applications that communicate via UDP/IP, some routines analogous to those of the Network Layer are provided. These routines manage the port assignments in the same way that the Network Layer routines manage the task name assignments. (Note that UDP communication that uses the Acnet or Classic protocols does not need to use these routines. They are automatically managed using the Network Layer; in a sense, the Network Layer routines operate at a higher level.) The UDP routines are designed to be used by the server programs that accept UDP datagrams containing either Acnet or Classic protocol messages. They can also be used by user applications that want to communicate UDP datagrams that contain other protocols.

```
UDPCnct(portReq: Integer; qId: Longint; evtMask: Integer;  
        VAR portId: Integer): Integer;
```

The first argument of this function is a UDP port# to be assigned. If it is zero on entry, a dynamic assignment is requested, and the newly-assigned index is written to the portId variable, unless none is available. If the value for portReq on entry is nonzero, it is a request for assignment of the given port#. If the given port# is already assigned, UDPCnct is first called to close the previous connection.

The qId is the message queue used for receiving messages directed to the assigned port#. The evtMask, if nonzero, specifies the bit mask used to signal the calling task of the arrival of a message.

```
UDPCnt(portId: Integer): Integer;
```

Remove the assignment of the given port# from the port table. The message queue field is cleared to indicate that the entry is available.

```
UDPQueue(portId: Integer; VAR mBlk: MBlkType;  
        VAR xmitStat: Integer): Integer;
```

Queue the message contained in the message block to the network and flush the network queue. The message block is a special type \$0016 used for UDP messages. NetXmit will precede the message with the IP and UDP headers formed from the destination node# within the message block.

```
UDPCheck(portId: Integer; timeOut: Longint;  
        VAR mRef: MRefType): Integer;
```

Check for a message waiting in message queue used by the given portId. A timeOut value of -1 specifies "no wait." A positive value is in 100 Hz ticks.

```
UDPRecv(VAR mRef: MRefType; VAR buf: BufType; maxSize: Integer): Integer;
```

Copy the message referenced by the mRef structure into the given buffer.

```
UDPOpen(port: Integer): Integer;
```

Establish an entry for given port# in the port table. A message queue is automatically created using the given port#. If the argument is zero, a dynamic assignment is made. The value of the function result is the portId. In this case, a zero result is an error, as the portId should be positive.

```
UDPClose(portId: Integer): Integer;
```

Close UDP port indicated by portId. This calls UDPDcnt and also deletes the message queue used by the connected port.

```
UDPRead(portId: Integer; VAR buf: BufType; mxSize: Integer;  
        VAR mSize: Integer; VAR srcN: Integer): Integer;
```

Check the message queue used by the given portId. If a message is present, return a copy of the message, along with its size and source node#.

```
UDPWrite(portId: Integer; VAR buf: BufType; size: Integer;  
        node: Integer; VAR xmitStat: Integer): Integer;
```

Write the message to the network. The function NetQueue is called after preparing the message block using the buffer contents. The node# is assigned from InsIPARP and specifies the info needed to fill the IP and UDP headers.