```
Mictcp_socket(
        return   Initialize_components(SERVEUR)
}

Int mic_tcp_bind(int socket,mic_tcp_sock_addr addr){
        If Mic_tcp_sock.addr ==Null{
                Mic_tcp_sock.fd = socket;
                Mic_tcp_sock.state =IDLE;
                Mic_tcp_sock.addr = addr;
                Return(0)
        }
        Else{
                Return(-1)
        }
}

Mic_tcp_accept(int socket,mic_tcp_scok_addr* addr){
        While(PDU ==0){
                IP_recv(PDU,@autre,machine,timeout)
        }
        If(PDU.header.syn==1){
                NewPDU.header.ack =1
                NEWPdu.header.syn = 1
                Construction addresse
                IP_send(NewPDU,@autremachine)
                IP_recv(PDU,@autremachine,timeout)
                If(PDU.header.ack == 1){
                        Etat connecté.
                        Return(0)
                }
                Else{
```

```
                        Return(-1)
                }
        }
        Else{
                Return(-1)
        }
}

Mictcp_connect{
        PDU.header.syn = 1
        IP_Send(PDU,@dest)
        IP_recv(PDU,@autre_machine,timeout)
        If(PDU.header.ack ==1 & PDU.header.syn == 1)
        {
                NewPDU.header.ack=1
                IP_Send(NewPDU,@dest)
                Etat connecté == 1
                Return(0)
        }
        Else{
                Return(-1)
        }
}

Mictcp_send( id, message, taille)
{
        PDU.payload
        PDU.Header.ack num = PE
        IP_Send(PDU,@dest)
        PE++
}
```

```
Process_received_PDU(PDU, addr){
        If(PDU.header.num_seq == PA){
        PA++
         Return App_buffer_put(PDU.payload)
}

Mic_tcp_recv(socket, mesg,max_mesg_size){
        Payload.dat = message
        Return(app_buffer_get(payload))
}
```