...
...

Process Model Attributes			
attribute	*value*	*type*	*default value*
addr_file	promoted	typed file	NONE

Header Block

```
#include "ip_arp_sup.h"
#include <sys/file.h>

typedef struct
    {
    int             node_num;
    int             phys_addr;
    } arp_entry;


/* Transition Mnemonics */
#define ARRIVAL      op_intrpt_type () == OPC_INTRPT_STRM

/* identification constant for IP protocol within */
/* Ethernet frames */
#define NET_PROT_IP      0x800
```

State Variable Block

```
arp_entry*          \addr_table;
int                 \num_entries;

char                \addr_file [512];

Objid               \my_id;
Ici*                \mac_iciptr;

arp_table*          \at_ptr;
```

Temporary Variable Block

```
int         instrm, next_node;
char        str0 [512], str1 [512];
Packet*     pkptr;
Ici*        iciptr;
int         phys_addr;
int         status;
char        error_string [512];
```

forced state **INIT**			
attribute	*value*	*type*	*default value*
name	INIT	string	st
enter execs	(See below.)	textlist	(See below.)
exit execs	(empty)	textlist	(empty)
status	forced	toggle	unforced

...
...

enter execs **INIT**

```
/* Obtain the object ID of the surrounding ARP processor. */
my_id = op_id_self ();

/* Obtain the name of the address conversion file */
/* to be used by this ARP processor. */
op_ima_obj_attr_get (my_id, "addr_file", addr_file);

/* Load the conversion table into memory. */
at_ptr = arp_sup_table_setup (addr_file);

/* Create an Ici to communicate with layer 2. */
mac_iciptr = op_ici_create ("ip_mac_req");
```

transition **INIT -> WAIT**

attribute	*value*	*type*	*default value*
name		string	tr
condition		string	
executive		string	
color	RGB333	color	RGB333
drawing style	spline	toggle	spline

unforced state **WAIT**

attribute	*value*	*type*	*default value*
name	WAIT	string	st
enter execs	(See below.)	textlist	(See below.)
exit execs	(empty)	textlist	(empty)
status	unforced	toggle	unforced

enter execs **WAIT**

```
```

transition **WAIT -> ARRIVAL**

attribute	*value*	*type*	*default value*
name		string	tr
condition	ARRIVAL	string	
executive		string	
color	RGB333	color	RGB333
drawing style	spline	toggle	spline

...
...

forced state **ARRIVAL**			
attribute	*value*	*type*	*default value*
name	ARRIVAL	string	st
enter execs	(See below.)	textlist	(See below.)
exit execs	(empty)	textlist	(empty)
status	forced	toggle	unforced

enter execs **ARRIVAL**

```
   /* An IP packet has arrived; determine on which input. */
   instrm = op_intrpt_strm ();

   /* Obtain the packet and the accompanying ICI */
 5 pkptr = op_pk_get (instrm);
   iciptr = op_intrpt_ici ();
   if (iciptr == OPC_NIL)
        {
        sprintf (error_string, "Simulation aborted; error in object (%d)",
10           op_id_self ());
        op_sim_end (error_string, "ip_arp: required ICI not received", " ", " ");
        }

   /* Obtain the next node number. */
15 op_ici_attr_get (iciptr, "next_node", &next_node);

   /* Determine the corresponding physical address. */
   status = arp_sup_phys_addr_get (at_ptr, next_node, &phys_addr);

20 /* If the address translation failed, issue an error message. */
   if (status == OPC_COMPCODE_FAILURE)
        {
        sprintf (str0, "ARP: discarding unmappable packet (%d)", op_pk_id (pkptr));
        sprintf (str1, "node number: (%d)", next_node);
25      op_pk_destroy (pkptr);
        op_sim_message (str0, str1);
     }

   else{
30      /* place the destination address and protocol type (IP) into the ICI */
        op_ici_attr_set (mac_iciptr, "dest_addr", phys_addr);
        op_ici_attr_set (mac_iciptr, "protocol_type", NET_PROT_IP);

        /* send the packet coupled with the ICI */
35      op_ici_install (mac_iciptr);
        op_pk_send (pkptr, 0);
        }
```

transition **ARRIVAL -> WAIT**			
attribute	*value*	*type*	*default value*
name		string	tr
condition		string	
executive		string	
color	RGB333	color	RGB333
drawing style	spline	toggle	spline

...
...