...
...

Process Model Attributes

attribute	*value*	*type*	*default value*
NGRP	promoted	integer	0

Header Block

```
#include "ip_frag_sup.h"

#define ARRIVAL_FRM_IP      op_intrpt_type () == OPC_INTRPT_STRM
#define UPD_INT             ((op_intrpt_type () == OPC_INTRPT_SELF) && (op_intrpt_code () == 0))
#define SH_INT              ((op_intrpt_type () == OPC_INTRPT_SELF) && (op_intrpt_code () == 1))

#define MAX_NO_LINKS        10
#define MAX_NO_GRPS         50

#define QOSPF_STREAM        1
#define IP_STREAM           0

#define ON                  1
#define OFF                 0

#define VL                  if(ON)
#define IGMPDL              if(OFF)

#define IS_MULTICAST        500

typedef struct{
	int interface[MAX_NO_LINKS];
	int mtu[MAX_NO_LINKS];
}mc_id;

typedef struct{
	int me[MAX_NO_GRPS];
}member;

extern int IGMP;
extern int UDP;
```

State Variable Block

```
Objid               \my_id,\ip_objid, \qospf_objid;
int                 \net_num0, \node_num0;
int                 \net_num1, \node_num1;
int                 \net_num2, \node_num2;
int                 \net_num3, \node_num3;
int                 \net_num4, \node_num4;
int                 \query_sent[100][5],\ngrp;
double              \upd;
ip_dgram_list*      \igmp_list_ptr;
int                 \grp_sent;
int                 \n_links;
```

...
...

Temporary Variable Block

```
int        xx,disp,j,i,proto,ip_strm,in_strm,join,term,grp_id,data_len;
Packet     *tmp_pk1,*tmp_pk2,*igmp_pkptr,*ip_pkptr,*decap_pkptr,*pkptr;
int     *tmp_ptr;
mc_id      *mc_table;
int        SEND_RESP,dest_node,src_net,src_node,Indep_Rou;
Objid      sub_netid;
```

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

```
grp_sent = 0;
my_id = op_id_self();
op_ima_obj_attr_get(my_id,"NGRP", &ngrp);
ip_objid = op_topo_assoc( my_id, OPC_TOPO_ASSOC_OUT, OPC_OBJMTYPE_MODULE, 0 );
qospf_objid = op_topo_assoc( my_id, OPC_TOPO_ASSOC_OUT, OPC_OBJMTYPE_MODULE, QOSPF_STREAM );

 /* Use the object id to get the address */
op_ima_obj_attr_get (ip_objid, "net_num0", &net_num0);
op_ima_obj_attr_get (ip_objid, "node_num0", &node_num0);
op_ima_obj_attr_get (ip_objid, "net_num1", &net_num1);
op_ima_obj_attr_get (ip_objid, "node_num1", &node_num1);
op_ima_obj_attr_get (ip_objid, "net_num2", &net_num2);
op_ima_obj_attr_get (ip_objid, "node_num2", &node_num2);
op_ima_obj_attr_get (ip_objid, "net_num3", &net_num3);
op_ima_obj_attr_get (ip_objid, "node_num3", &node_num3);
op_ima_obj_attr_get (ip_objid, "net_num4", &net_num4);
op_ima_obj_attr_get (ip_objid, "node_num4", &node_num4);
op_ima_obj_attr_get (ip_objid, "Indep_Rou", &Indep_Rou);
op_intrpt_schedule_self(op_sim_time() + 6, 0);
/*op_intrpt_schedule_self(op_sim_time() + 5.9, 1);*/

for(i = 0; i < MAX_NO_GRPS; ++i)
{
	for(j = 0; j < 5; ++j)
		query_sent[i][j] = 0;
}

if(!Indep_Rou)
	sub_netid=op_topo_parent(op_topo_parent(ip_objid));
else
	sub_netid=op_topo_parent(ip_objid);

n_links = op_topo_assoc_count(sub_netid,OPC_TOPO_ASSOC_IN,OPC_OBJTYPE_LKSIMP);

VL  IGMPDL printf("N-links at %d.%d = %d\n",net_num0,node_num0,n_links);
```

...
...

transition **init -> idle**			
attribute	*value*	*type*	*default value*
name	tr_32	string	tr
condition		string	
executive		string	
color	RGB333	color	RGB333
drawing style	spline	toggle	spline

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

enter execs **idle**

transition **idle -> FRM_IP**			
attribute	*value*	*type*	*default value*
name	tr_1	string	tr
condition	ARRIVAL_FRM_IP	string	
executive		string	
color	RGB333	color	RGB333
drawing style	spline	toggle	spline

transition **idle -> UPD**			
attribute	*value*	*type*	*default value*
name	tr_7	string	tr
condition	UPD_INT	string	
executive		string	
color	RGB333	color	RGB333
drawing style	spline	toggle	spline

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

enter execs **FRM_IP**

```
in_strm = op_intrpt_strm();
decap_pkptr = op_pk_get (in_strm);
op_pk_nfd_get(decap_pkptr,"src_net",&src_net);
op_pk_nfd_get(decap_pkptr,"src_node",&src_node);
```

...
...

```
op_pk_nfd_get(decap_pkptr,"strm",&ip_strm);
op_pk_nfd_get(decap_pkptr,"prot_id",&proto);
op_pk_nfd_get(decap_pkptr,"data",&igmp_pkptr);
op_pk_nfd_get(decap_pkptr,"dest_node",&dest_node);
op_pk_nfd_get(igmp_pkptr,"terminate",&term);
op_pk_nfd_get(igmp_pkptr,"GRP_ID",&grp_id);
op_pk_nfd_get(igmp_pkptr,"join",&join);

if(proto != IGMP)
    op_sim_end("IGMP in router is getting the wrong pks","","","");
if(join == 1 && term == 0)
{
    int Send_Qospf_Flag = OFF;
    mc_table = (mc_id *) op_ima_obj_svar_get(ip_objid,"mc_table");
    query_sent[grp_id][ip_strm] = 0;

    if(mc_table[grp_id].interface[ip_strm] != 1 )
        Send_Qospf_Flag = ON;
        mc_table[grp_id].interface[ip_strm] = 1;
    op_pk_destroy(decap_pkptr);

    if(Send_Qospf_Flag)
        op_intrpt_schedule_remote( op_sim_time(), 6, qospf_objid );
        /* op_pk_send( igmp_pkptr, QOSPF_STREAM );    */
    /* else   */
    op_pk_destroy(igmp_pkptr);
}

else
{
    if( dest_node == IS_MULTICAST )
    {
        mc_table = (mc_id *) op_ima_obj_svar_get(ip_objid,"mc_table");
        tmp_pk1 = op_pk_copy(igmp_pkptr) ; tmp_pk2 = op_pk_copy(decap_pkptr);
        for(j=0;j< ngrp +1; j++)
        {
            igmp_pkptr = op_pk_copy(tmp_pk1);
            decap_pkptr = op_pk_copy(tmp_pk2);
            SEND_RESP = 0;
            i = 0;
            while(i < n_links + 1  && !SEND_RESP)
            {
                if(mc_table[j].interface[i] == 1 && i != ip_strm) SEND_RESP = 1;
                i++;
            }
            if(src_net == net_num0 && src_node == node_num0) SEND_RESP = 0;
            if(SEND_RESP)
            {
                op_pk_nfd_set(igmp_pkptr, "join", 1);
                op_pk_nfd_set(igmp_pkptr, "terminate", 0);
                op_pk_nfd_get(decap_pkptr, "src_net", &src_net);
                op_pk_nfd_get(decap_pkptr, "src_node", &src_node);
                op_pk_nfd_set(decap_pkptr, "dest_net", src_net);
                op_pk_nfd_set(decap_pkptr, "dest_node", src_node);
                op_pk_nfd_set(decap_pkptr, "src_net", net_num0);
                op_pk_nfd_set(decap_pkptr, "src_node", node_num0);
                op_pk_nfd_set(igmp_pkptr, "GRP_ID", j);
                op_pk_nfd_set(decap_pkptr, "data", igmp_pkptr);
                op_pk_nfd_set(decap_pkptr, "ttl", 2);
```

```
                        op_pk_send(decap_pkptr, IP_STREAM);
                    }
                    else
                    {
                        op_pk_destroy(decap_pkptr);
                        op_pk_destroy(igmp_pkptr);
                    }
                }
                op_pk_destroy(tmp_pk1);
                op_pk_destroy(tmp_pk2);
            }
        else op_sim_end("Error in ROU_IGMP !!","","","");
}
```

exit execs **FRM_IP**

```
/*mc_table = (mc_id *) op_ima_obj_svar_get(ip_objid,"mc_table");

for(i=0;i<10;++i)
  {  j=0;
     while(j<5)
       { if( mc_table[i].interface[j] == 1)
              printf("Member of grp: %d on interface %d (%d.%d)\n",i,j,net_num0,node_num0);
         j++;
       }
  }
*/
```

transition **FRM_IP -> idle**			
attribute	*value*	*type*	*default value*
name	tr_3	string	tr
condition		string	
executive		string	
color	RGB333	color	RGB333
drawing style	spline	toggle	spline

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

enter execs **UPD**

```
mc_table = (mc_id *) op_ima_obj_svar_get(ip_objid,"mc_table");
disp = 0;
i=0;
while(i < ngrp +1)
{
        for(j=0;j < n_links+1;++j)
        {
                if(mc_table[i].interface[j] != 1)
```

...
...

```
                    query_sent[i][j] == 0;
            else
                    query_sent[i][j]++;
            if(query_sent[i][j] == 2 && mc_table[i].interface[j] == 1)
            {
                    disp = 1;
                    printf("Tearing down for grp: %d on intrface %d at %d.%d\n",i,j,net_num0,node_num0);
                    mc_table[i].interface[j] = 0;
            }
        }
        i++;
}

pkptr = op_pk_create_fmt("MC_APP_CNTRL");
op_pk_nfd_set(pkptr,"join",0);
op_pk_nfd_set(pkptr,"terminate",0);
op_pk_nfd_set(pkptr,"GRP_ID",500);

data_len = op_pk_total_size_get (pkptr) / 8;
ip_pkptr = op_pk_create_fmt ("ip_dgram");
op_pk_nfd_set (ip_pkptr, "data", pkptr);
op_pk_nfd_set (ip_pkptr, "prot_id", IGMP);
op_pk_bulk_size_set (ip_pkptr, data_len * 8);

op_pk_nfd_set (ip_pkptr, "dest_net", 500);
op_pk_nfd_set (ip_pkptr, "dest_node", 500);

/* Set the source address */
op_pk_nfd_set (ip_pkptr, "src_net", net_num0);
op_pk_nfd_set (ip_pkptr, "src_node", node_num0);

/* set the length attributes of the IP packet. */
op_pk_nfd_set (ip_pkptr, "orig_len", data_len);
op_pk_nfd_set (ip_pkptr, "frag_len", data_len);
op_pk_nfd_set (ip_pkptr, "ttl",2);

/* Indicate that the packet is not yet fragmented. */
op_pk_nfd_set (ip_pkptr, "frag", 0);
op_pk_send(ip_pkptr,0);

op_intrpt_schedule_self(op_sim_time()+60,0);

if(disp)
{
        mc_table = (mc_id *) op_ima_obj_svar_get(ip_objid,"mc_table");
        printf("Time: %f       TEARING  IP Addr: %d.%d\n",op_sim_time(),net_num0,node_num0);
        printf("GRP   Interface   \n");
        for(i=0;i< ngrp+1;++i)
        {
            j=0;
            while(j<5)
            {
                    if( mc_table[i].interface[j] == 1)
                    printf("%6d%12d\n",i,j);
                    j++;
            }
        }
}
```

...
...

exit execs UPD

```
/*mc_table = (mc_id *) op_ima_obj_svar_get(ip_objid,"mc_table");
printf("%f:      GRP Interface        IP Addr\n",op_sim_time());
for(i=0;i<10;++i)
  {  j=0;
     while(j<5)
       { if( mc_table[i].interface[j] == 1)
              printf("                        %d   %d   (%d.%d)\n",i,j,net_num0,node_num0);
         j++;
       }
  }
getchar();*/
```

transition UPD -> idle

attribute	*value*	*type*	*default value*
name	tr_8	string	tr
condition		string	
executive		string	
color	RGB333	color	RGB333
drawing style	spline	toggle	spline

forced state dis

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

enter execs dis

```
/*mc_table = (mc_id *) op_ima_obj_svar_get(ip_objid,"mc_table");
printf("Time: %f                JOINING Ip Addr: %d.%d\n",op_sim_time(),net_num0,node_num0);
printf("GRP   Interface   \n");
for(i=0;i< ngrp+1;++i)
  {  j=0;
     while(j<5)
       { if( mc_table[i].interface[j] == 1)
              printf("%6d%13d\n",i,j);
         j++;
       }
  }
getchar();
*/
```

exit execs dis

```
op_intrpt_schedule_self(op_sim_time() + 60, 1);
```

...
...

transition **dis -> idle**

attribute	*value*	*type*	*default value*
name	tr_29	string	tr
condition		string	
executive		string	
color	RGB333	color	RGB333
drawing style	spline	toggle	spline