...
...

Process Model Attributes

attribute	*value*	*type*	*default value*
NGRPS	promoted	integer	0
SESSMIN	promoted	double	0.0 (sec)
SESSMAX	promoted	double	0.0 (sec)
DATA_IAT	promoted	double	0.0 (sec)
SESS_IAT	promoted	double	0.0 (sec)

Header Block

```
#define   PK_ARR op_intrpt_type () == OPC_INTRPT_STRM
#define   SELF_INT0   (op_intrpt_type () == OPC_INTRPT_SELF  && op_intrpt_code () == 0)
#define   SELF_INT1   (op_intrpt_type () == OPC_INTRPT_SELF  && op_intrpt_code () == 1)

#define END_SIM        op_intrpt_type () ==  OPC_INTRPT_ENDSIM

#define ON                  1
#define OFF                 0

#define VL                  if(ON)
#define MADL                if(ON)

#define IGMP_STREAM     0
#define IP_STREAM       1

#define UDP_PROTOCOL_ID     20

typedef struct {
        char Message[500];
}App_Data;

typedef struct{
        int NetNum;
        int NodeNum;
}IPAddress;
```

State Variable Block

```
Objid           \my_id, \ip_objid;
int             \grp_id,\sess;
double          \tim,\data_rec;

int \dest_net, \dest_node, \data_len;

int     \ngrps;
Distribution* \grp_dist,\time_dist,\term_dist;
double     \sessmin,\sessmax,\data_iat,\sess_iat;

IPAddress \LocalAddress;
```

Temporary Variable Block

```
Packet*  pkptr;
int in_strm,da1,da2,da3;
```

```
FILE*  fil;
```

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**

```
my_id = op_id_self();

ip_objid = op_topo_assoc (my_id, OPC_TOPO_ASSOC_OUT, OPC_OBJMTYPE_MODULE, IP_STREAM);

op_ima_obj_attr_get (ip_objid, "net_num0", &LocalAddress.NetNum );
op_ima_obj_attr_get (ip_objid, "node_num0", &LocalAddress.NodeNum );

data_rec = 0;

op_ima_obj_attr_get(my_id,"NGRPS",&ngrps);
op_ima_obj_attr_get(my_id,"SESSMIN",&sessmin);
op_ima_obj_attr_get(my_id,"SESSMAX",&sessmax);
op_ima_obj_attr_get(my_id,"DATA_IAT", &data_iat);
op_ima_obj_attr_get(my_id,"SESS_IAT",&sess_iat);

/*generate a RV between 0 and 99 (max of 100 grps to
select which grp id to join*/
grp_dist = op_dist_load("uniform_int",0,ngrps);

/*time when he joins initially*/
time_dist = op_dist_load("exponential",sess_iat,0);
tim = op_dist_outcome(time_dist);
op_intrpt_schedule_self((op_sim_time() + tim/100 + 1),0);
sess = 0;

/*op_ima_obj_attr_get (op_id_self (), "GRP",&grp_id);*/

term_dist = op_dist_load("uniform",sessmin,sessmax);
```

transition **init -> idle**			
attribute	*value*	*type*	*default value*
name	tr_0	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	(empty)	textlist	(empty)
exit execs	(empty)	textlist	(empty)
status	unforced	toggle	unforced

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

transition **idle -> JOIN_TERM**			
attribute	*value*	*type*	*default value*
name	tr_4	string	tr
condition	SELF_INT0	string	
executive		string	
color	RGB333	color	RGB333
drawing style	spline	toggle	spline

transition **idle -> data**			
attribute	*value*	*type*	*default value*
name	tr_6	string	tr
condition	SELF_INT1	string	
executive		string	
color	RGB333	color	RGB333
drawing style	spline	toggle	spline

transition **idle -> END**			
attribute	*value*	*type*	*default value*
name	tr_9	string	tr
condition	END_SIM	string	
executive		string	
color	RGB333	color	RGB333
drawing style	spline	toggle	spline

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

enter execs **pk_arr**

```
pkptr  = op_pk_get( op_intrpt_strm () );

op_pk_nfd_get( pkptr, "join", &da1 );
```

...
...

```
op_pk_nfd_get( pkptr, "terminate", &da2 );
op_pk_nfd_get( pkptr, "GRP_ID", &da3 );

VL  MADL
{
    if(da1 == 0 && da2 == 0)
        printf("ONLY DATA REACHES APP GRP_ID: %d\n AND REGISTERED IN %d",da3,grp_id);
    else
        printf("Something other than data has come here\n");

    printf("\nIam a member of %d and got a %d pk\n",grp_id,da3);
}

data_rec++;
op_pk_destroy( pkptr );
```

transition **pk_arr -> idle**			
attribute	*value*	*type*	*default value*
name	tr_2	string	tr
condition		string	
executive		string	
color	RGB333	color	RGB333
drawing style	spline	toggle	spline

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

enter execs **JOIN_TERM**

```
if( !sess )
{
    /*
    ** Join into a session  if the application is not currently in any multicast
    ** session. Schedule an interupt for terminating this session. Also schedule
    ** an interrupt for  sending data to this session. If you  are a application
    ** that  do not want to send any data donot schedule the interrupt 1 ( which
    ** is the data send interrupt)
    */
    grp_id =  op_dist_outcome(grp_dist);

    pkptr = op_pk_create_fmt("MC_APP_CNTRL");

    op_pk_nfd_set( pkptr,"join",1 );
    op_pk_nfd_set( pkptr,"terminate",0 );
    op_pk_nfd_set( pkptr,"GRP_ID",grp_id );
    op_pk_send( pkptr, IGMP_STREAM );
    VL  MADL
    {
        printf("Sending a join pk from Application \n");
    }
```

```
        sess=1;

        /*Interrupt for sending data pks to the session */
        op_intrpt_schedule_self( (op_sim_time() + data_iat), 1 );

        /* Interrupt for terminating the existing session */
        tim = op_dist_outcome(term_dist);
        op_intrpt_schedule_self( (op_sim_time() + tim), 0 );
}

if(sess)
{
        /*
        ** Terminate the current existing session and schedule an interrupt for
        ** joining another session
        */
        pkptr = op_pk_create_fmt("MC_APP_CNTRL");
        op_pk_nfd_set( pkptr,"join",0 );
        op_pk_nfd_set( pkptr,"terminate",1 );
        op_pk_nfd_set( pkptr,"GRP_ID",grp_id );
        op_pk_send( pkptr,IGMP_STREAM );
        sess--;

        tim = op_dist_outcome(time_dist);
        op_intrpt_schedule_self( (op_sim_time() + tim), 0 );
}
```

transition **JOIN_TERM -> idle**			
attribute	*value*	*type*	*default value*
name	tr_5	string	tr
condition		string	
executive		string	
color	RGB333	color	RGB333
drawing style	spline	toggle	spline

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

enter execs **data**

```
if( sess )
{
        /*
        ** If currently registered with a session, send an data packet to the session.
        ** Size of the data packet is 576 bytes. This data packet is encapsulated into
        ** an IP packet and sent to the IP layer. The Destiantion net in the IP packet
        ** is set  to the mulitcast address( 500 ) and the  destination node in the IP
        ** packet is the  Mulitcast Group ID of  the session that it is sending packet
        ** to. Interrupt is schedule for sending the next data packet.
        */
```

...
...

```
        App_Data DataSent;
        Packet* ip_pkptr;
        sprintf( DataSent.Message, "This was data sending to the host %d.%d by %d.%d \n",
            dest_net, dest_node, LocalAddress.NetNum, LocalAddress.NodeNum );
    pkptr = op_pk_create_fmt("Unicast_Appl");

    op_pk_nfd_set(pkptr, "UniData", &DataSent, op_prg_mem_alloc,
            op_prg_mem_free, sizeof(App_Data) );

    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", UDP_PROTOCOL_ID );
    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", grp_id );

        op_pk_nfd_set(ip_pkptr, "src_net", LocalAddress.NetNum );
    op_pk_nfd_set(ip_pkptr, "src_node", LocalAddress.NodeNum );

        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 );
    op_pk_nfd_set(ip_pkptr, "frag", 0 );

        op_pk_send(ip_pkptr, IP_STREAM);
}

/* Interrupt for the next data packet */
op_intrpt_schedule_self( (op_sim_time() + data_iat), 1 );
```

transition **data -> idle**			
attribute	*value*	*type*	*default value*
name	tr_8	string	tr
condition		string	
executive		string	
color	RGB333	color	RGB333
drawing style	spline	toggle	spline

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

enter execs **END**

```
fil = fopen("OP","a+");
fprintf(fil,"\nAve DATA recieved : %f pks/sec",data_rec/op_sim_time());
fclose(fil);
```

...
...