INTERNATIONAL ATOMIC ENERGY AGENCY
UNITED NATIONS EDUCATIONAL, SCIENTIFIC AND CULTURAL ORGANIZATION
INTERNATIONAL CENTRE FOR THEORETICAL PHYSICS
I.C.T.P., P.O. BOX 586, 34100 TRIESTE, ITALY, CABLE: CENTRATOM TRIESTE

H4.SMR. 405/17

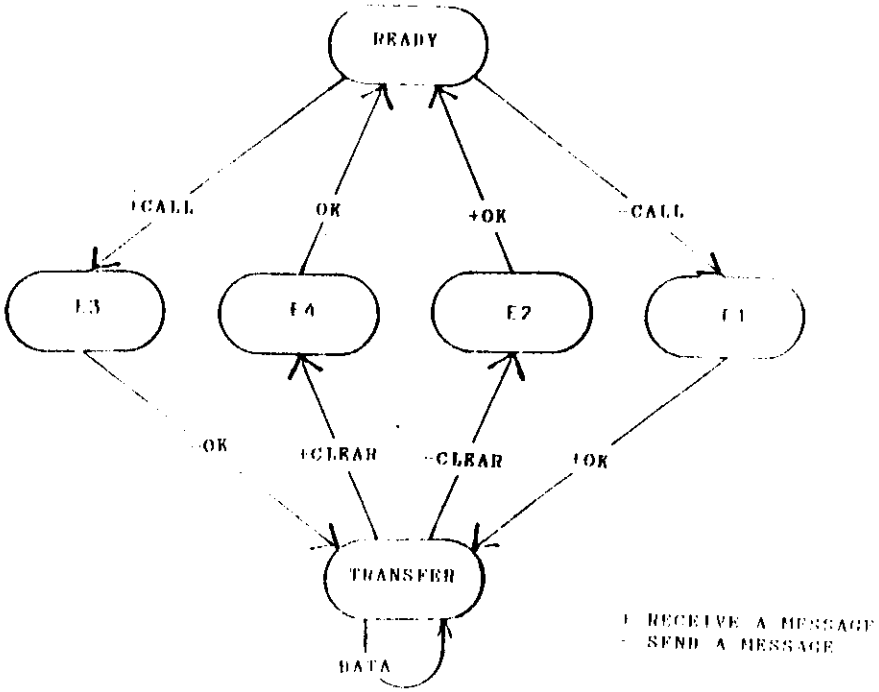
SECOND WORKSHOP ON TELEMATICS

6 - 24 November 1989

A Simple Communication Protocol

J. GROMPONE
Interfase Ltd., Montevideo, Uruguay

A SIMPLE COMMUNICATION PROTOCOL



PROTOCOL ABSTRACTION

```
type STATE is (READY, TRANSFER, E1, E2);

type MESSAGE is (VOID, CALL, CLEAR, OK,
                 DATA);

task ENTITY is
     entry DATA(M: in MESSAGE);
end ENTITY;

procedure MACHINE (
     E: in out STATE;
     M: in MESSAGE;
     R: out MESSAGE
      );

task body ENTITY is
E : STATE := READY;
S : MESSAGE;
R : MESSAGE;
begin
     loop
          accept DATA(M: in MESSAGE) do
               S := M;
          end DATA;
          MACHINE(E, S, R);
          ENTITY.DATA( R);
     end loop;
end ENTITY;
```

STATES TRANSITIONS MACHINE

```
procedure MACHINE( E: in out STATE;
                   M: in MESSAGE;
      R: out MESSAGE) is
begin
      case E is
            when READY =>
                  if M = CALL then
                        E := TRANSFER;
                        R := OK;
                  end if;

            when E1 =>
                  if M = OK then
                        E := TRANSFER;
                  end if;

            when TRANSFER =>
                  if M = CLEAR then
                        E := READY;
                        R := OK;
                  end if;

            when E2 =>
                  if M = OK then
                        E := READY;
                  end if;
      end case;
end MACHINE;
```


PROTOCOL WITH TIME OUT

```
function TIMEOUT(E: in STATE)
      return boolean;

function TIME(E: in STATE)
      return DURATION;

function STATE_TIME(E: in STATE)
      return STATE;

function ACTION(E: in STATE)
      return MESSAGE;


task body ENTITY is
E : STATE := READY;
S : MESSAGE;
R : MESSAGE;
begin
   loop
      select
            accept DATA(M: in MESSAGE) do
                  S := M;
            end DATA;
            MACHINE(E, S, R);
            ENTITY.DATA( R);
      or
            when TIMEOUT(E) =>
                  delay TIME(E);
                  R := ACTION(E);
                  E := STATE_TIME(E);
                  ENTITY.DATA( R);
      end select;
   end loop;
end ENTITY;
```

```
type MESSAGE_N       is ( VOID, CALL, CLEAR,
                          OK, DATA);
type MESSAGE_N_UP    is ( VOID, ... );
type MESSAGE_N_INF is ( VOID, ...);

type STATE_N is (READY, TRANSFER, E1, E2,
                 E3, E4);

task ENTITY_N is
   entry LAYER_UP( M: in MESSAGE_N_UP);
   entry LAYER_DOWN( M: in MESSAGE_N_INF);
end ENTITY_N;

task ENTITY_N_UP is
   entry LAYER_UP( M: in MESSAGE ...);
   entry LAYER_DOWN( M: in MESSAGE_N);
end ENTITY_N_UP;

task ENTITY_N_INF is
   entry LAYER_UP( M: in MESSAGE_N);
   entry LAYER_DOWN( M: in MESSAGE ...);
end ENTITY_N_INF;

procedure MACHINE_UP(
                 E: in out  STATE_N;
                 M_AUX: in  MESSAGE_N_UP;
                 M_UP:  out MESSAGE_N_UP;
                 M_INF: out MESSAGE_N_INF
                 );

procedure MACHINE_INF(
                 E:  in out STATE_N;
                 M_AUX: in  MESSAGE_N_INF;
                 M_UP: out  MESSAGE_N_UP;
                 M_INF: out MESSAGE_N_INF
                 );

procedure MACHINE_TIMES(
                 E: in out  STATE_N;
                 V: in out integer;
                 M_UP: out  MESSAGE_N_UP;
                 M_INF: out MESSAGE_N_INF
                 );
```



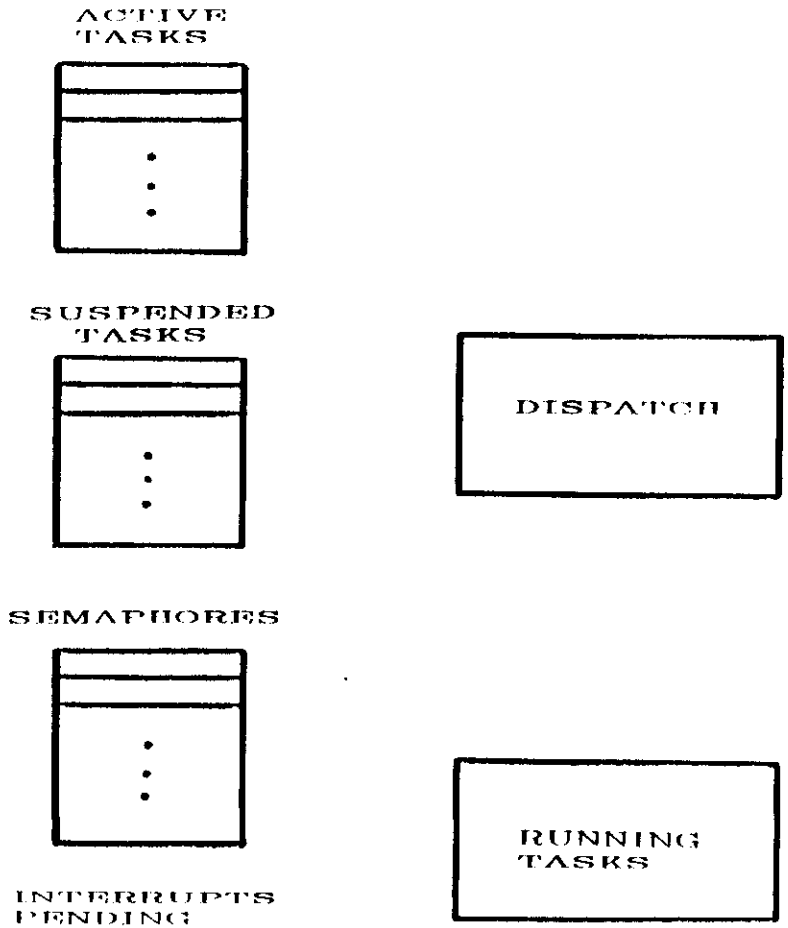
```
task body ENTITY_N is
E:       STATE_N := READY;
M_UP:    MESSAGE_N_UP;
M_INF:   MESSAGE_N_INF;
V:       integer := 0;

begin
  loop
     select
        accept LAYER_UP
                ( M: in MESSAGE_N_UP ) do
           M_UP := M;
        end LAYER_UP;
        -- update state
        MACHINE_UP(E, M_UP, M_UP, M_INF);
        -- send messages to the layers
        ENTITY_N_UP.LAYER_DOWN(M_UP);
        ENTITY_N_INF.LAYER_UP(M_INF);
     or
        accept LAYER_DOWN
                ( M: in MESSAGE_N_INF) do
           M_INF := M;
        end LAYER_DOWN;
        -- update state
        MACHINE_INF(E, M_INF, M_UP, M_INF)
        -- send messages to the layers
        ENTITY_N_UP.LAYER_DOWN(M_UP);
        ENTITY_N_INF.LAYER_UP(M_INF);
     or
        when TIMEOUT(E) =>
           delay TIME(E);
           -- update state
           MACHINE_TIMES
             (E, V, M_UP, M_INF);
           -- send messages to the layers
           ENTITY_N_UP.LAYER_DOWN(M_UP
           ENTITY_N_INF.LAYER_UP(M_INF);
     end select;
  end loop;
end ENTITY_N;
```

REAL-TIME NUCLEOUS

DISPATCH PROCESS:

- HANDLE INTERRUPTS
- BLOCK / WAKE UP PROCESSES
- PRIORITY ASSIGNMENT
- REAL-TIME ACTIVITIES

DISPATCHER



IMPLEMENTATION LANGUAGES

CONDITIONS:

- A GOOD STRUCTURE FOR DATA AND ALGORITHMS
- SUPPORT MODULARITY
- STRONG TYPED
- SUPPORT STAND ALONE PROGRAM CREATION
- SUPPORT ROM PROGRAM CREATION
- FREE ALLOCATION OF SEGMENTS

ASSEMBLER:

- VERY COMPACT CODES
- EFFICIENT IN TIME
- DIFFICULT TO UNDERSTAND
- DIFFICULT TO BE RELIABLE

C:

- POORLY TYPED
- FULL OF SIDE EFFECTS
- USUALLY EFFICIENT:

	ASSEMBLER	C
CODE	2K	8K
TIME	2ms	6–20 ms

SOME C EXAMPLES

++ OPERATOR:

```
i++
printf("%d\n",++i)
printf("%d\n",i++)
```

THE FUNCTION ASIGNATION:

```
value of (a = 5) is 5
a = b = c = 5;
a = ( b = ( c = 1 ));
```

IF CLAUSE:

```
if (a = 1) is always true
if (a == 1)
```

ONLY FUNCTIONS:

```
func(a,&b,&c)
```

NOT STRONGLY TYPED:

```
char a
int  i
i = a * 5
```

CRYPTIC SOME TIMES:

```
for(i=1; i<10; printf("%d\n",i++))
for(;;)
```

CHILL

1980: CCITT RECOM. Z.200

- STRONG TYPED
- "SIMILAR" TO PASCAL
  - MODULES
- REPRESENTATION CLAUSES
- REAL TIME SUPPORT
  - EXCLUDED VARIABLES
  - MAILBOXES
  - SIGNALS
- VERY FEW COMPILERS

ADA

1983: AMERICAN DEPT. OF DEFENSE

- STRONG TYPED
- PASCAL LIKE
  - MODULES (PACKAGES)
  - REPRESENTATION CLAUSES
- REAL-TIME SUPPORT
- GENERIC UNITS
- COMPILERS MUST BE VALIDATED BY D.O.D.