Use of the Transputer Event Line from Helios

Perihelion Software Technical Report No. 13

Andy Evans

January 1989

Perihelion Software Limited
The Maltings
Charlton Road
Shepton Mallet
Somerset
BA4 5QE
England
Telephone +44 749 4203
Fax. +44 749 4977

Contents

Whenever the transputer's event input is asserted, Helios responds by executing each of the handlers which have been added into the kernel's list. This technical note describes the Helios routines, SetEvent() and RemEvent(), which are used to add or remove the handler, and the event structure which is used to define the handler. An example C program is included to show how a simple handler is used.

1 The Event structure

The event structure is used to define the event handler. It is defined as:

```
struct Event
{
  struct Node Node;
  word Priority;
  void (*Handler)();
  word *Data;
  word Reserved;
};
```

The instance of struct Node is used to link this event handler into a list of handlers which is maintained by the kernel. The position of the handler within this list is determined by the Priority field; small numbers represent high priority handlers and large numbers represent lower priority ones. The handler field is used to store a pointer to the handler's code, and the data field contains a pointer to its data area; this data pointer is passed as an argument to the handler routine whenever it is called.

2 SetEvent()

This routine should be used to install an event handler within the kernel's list of handlers. The syntax of this routine is:

```
word SetEvent( Event *handler )          /* returns an error code */
```

When called, this routine uses the priority field in the Event structure to determine the handler's position in the list. The handler will always be inserted before other handlers that have a higher value in their priority field.

3 RemEvent()

This routine removes the specified event-handler from the kernel's handler list. The syntax of this routine is:

```
word RemEvent( Event *handler )          /* returns an error code */
```

4 An Example Program

```
/* Program to demonstrate the action of SetEvent and           *
 * RemEvent, plus the calling convention of the event routine. */

#include <syslib.h>
#include <event.h>
#include <sem.h>

/* these two templates should be in event.h, but aren't. */
word SetEvent(Event *event):
word RemEvent(Event *event);

void do_something(word *data, Event *event);

Event event = {                 /* setup Event structure */
   NULL,NULL,
   0,
   do_something,
   NULL,
   NULL
}

Semaphore evsem;                /* event to process signal */

int main()
{
   InitSemaphore(&evsem,0);
   SetEvent(&event);
   Wait(&evsem);
   RemEvent(&event);
   return 0;
}

/* Event handier routine */
void do_something(word *data, Event *event)
{
   Signal(&evsem);
}
```