Use of Transputer Links from Helios

Perihelion Software Technical Report No. 5

N. Garnett

April 1989

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

Contents

1 Introduction

Most Helios programs will not need to explicitly reference the hardware links, as all message routing and multiplexing is transparently handled by the kernel. However, for high speed communications between adjacent processors, or for communicating with special hardware, it is often desirable to communicate directly via the links. This technical note shows how this is achieved.

2 Gaining Access to the Link

When explicitly communicating via links, it is not advisable to define a connection for them within the resource map. This is not essential, since any inter-processor link may be reconfigured. However if Helios is using the link for its own traffic it may take the Nucleus some time to detect that the link is no longer available and find another route. Obviously if the link is the only connection between two parts of the network it should not be reconfigured

Two operations are needed before the link is to be used. The first is to reconfigure the link to Dumb mode so Helios will not expect to send or receive messages through it; this is achieved by calling Configure to set the link mode. The second operation is to gain sole control of the link with AllocLink, which must eventually be matched by a call to FreeLink. Note that link configuration and allocation is not undone by Task termination; it is your responsibility to restore them to their original state.

3 Link Data Transfers

Once the link has been reconfigured and allocated successfully it may be used for data transfer. The Kernel provides two routines, LinkIn and LinkOut, for this purpose. Both take the link number, a buffer, the buffer's size and a timeout, and attempt the appropriate link transfer. The result is an error code which may indicate that the transfer was unsuccessful; possible causes are link not allocated or reconfigured, and transfer timeout. Note that the timeout uses the same mechanism as the message passing system, so it is not appropriate for very small grain timing.

Alternatively the Transputer I/O instructions may be used either from C or from Occam. The C header file asm.h contains macros for placing these instructions directly in C programs.

4 An Example

The following set of routines show how link I/O may be performed. The most important routines here are link_open and link_close. The other routines are merely jackets for LinkIn and LinkOut and are deliberately kept simple here; in practice you may want to be able to vary the timeouts and detect error which these example routines do not do.

```
#include <helios.h>     /* standard header           */
#include <link.h>       /* for templates & LinkInfo  */
#include <config.h>     /* for LinkConf              */
#include <codes.h>      /* for Err_Null              */

boot link_open(word linkno, LinkConf *save)
{
  LinkInfo linkinfo;
  LinkConf c;

  /* First save current link state */
  if( LinkData(linkno,&linkinfo) != Err_Null ) return FALSE;

  /* get the link into dumb mode */
  c.Id = linkno;            /* Link to change */
  c.Mode = Link_Mode_Dumb;  /* new mode       */
  c.State = 0;              /* unchanged      */
  c.Flags = 0;              /* unchanged      */

  /* change mode */
  if( Configure(c) != Err_Null ) return FALSE;

  /* get control of the link */
  if( AllocLink(linkno) != Err_Null ) return FALSE;

  /* Return the old configuration, the first four fields of a */
  /* LinkInfo structure match a LinkConf structure exactly so we */
  /* cheat a tittle here... */

  *save = *(LinkConf *)&linkinfo;
  return TRUE;
}

void link_close(LinkConf oldconf)
{
  /* relinquish control of the link */
  if( FreeLink(oldconf->Id) != Err_Null ) return;

  /* restore old state */
  if( Configure(oldconf) != Err_Null ) return;
```

```
}

void link_out_byte(word linkno, char b)
{
  LinkOut(1,linkno,&b,OneSec);
}

char link_in_byte(int linkno)
{
  char b = 0;
  LinkIn(1,linkno,&b,OneSec);
  return b;
}

void link_out_word(word linkno, word data)
{
  LinkOut(4,linkno,&data,OneSec);
}

word link_in_word(word linkno)
{
  word data = 0;
  LinkIn(4,linkno,&data,OneSec);
  return data;
}

void linkout_buf(word linkno, void *buf, word size)
{
  LinkOut(size,linkno,buf,OneSec);
)

void link_in_buf(word linkno, void *buf, word size)
{
  LinkIn(size,linkno,buf,OneSec);
}
```