```
DEFINITION MODULE ASCII ;

(*
   Author    : Gaius Mulley
   Date      : 12/12/85
   LastEdit  : 15/10/86
   Desciption: Defines all ascii constants (as in man ASCII)
               Note that lf, eof and EOL are added
*)

EXPORT QUALIFIED
     nul, soh, stx, etx, eot, enq, ack, bel,
     bs , ht , nl , vt , np , cr , so , si ,
     dle, dc1, dc2, dc3, dc4, nak, syn, etb,
     can, em , sub, esc, fs , gs , rs , us ,
     sp ,  (* All the above are in order *)
     lf, ff, eof, del, tab, EOL ;

CONST
     nul=000C; soh=001C; stx=002C; etx=003C;
     eot=004C; enq=005C; ack=006C; bel=007C;
     bs =010C; ht =011C; nl =012C; vt =013C;
     np =014C; cr =015C; so =016C; si =017C;
```



```
/* Header for module ASCII, generated by p2c */
#ifndef ASCIIH
#define ASCIIH


#ifdef ASCIIG
# define vextern
#else
# define vextern extern
#endif


/*
   Author    : Gaius Mulley
   Date      : 12/12/85
   LastEdit  : 15/10/86
   Desciption: Defines all ascii constants (as in man ASCII)
               Note that lf, eof and EOL are added
*/

/* All the above are in order */

#define ASCII_nul       '\0'
#define ASCII_soh       '\001'
#define ASCII_stx       '\002'
#define ASCII_etx       '\003'
```

```
    dle=020C; dc1=021C; dc2=022C; dc3=023C;
    dc4=024C; nak=025C; syn=026C; etb=027C;
    can=030C; em =031C; sub=032C; esc=033C;
    fs =034C; gs =035C; rs =036C; us =037C;
    sp =040C; (* All the above are in order *)
    lf =nl  ; ff =np  ; eof=eot ; tab=ht  ;
    del=177C; EOL=cr  ;

END ASCII.
```



```
#define ASCII_eot       '\004'
#define ASCII_enq       '\005'
#define ASCII_ack       '\006'
#define ASCII_bel       '\007'
#define ASCII_bs        '\b'
#define ASCII_ht        '\t'
#define ASCII_nl        '\n'
#define ASCII_vt        '\013'
#define ASCII_np        '\f'
#define ASCII_cr        '\015'
#define ASCII_so        '\016'
#define ASCII_si        '\017'
#define ASCII_dle       '\020'
#define ASCII_dc1       '\021'
#define ASCII_dc2       '\022'
#define ASCII_dc3       '\023'
#define ASCII_dc4       '\024'
#define ASCII_nak       '\025'
#define ASCII_syn       '\026'
#define ASCII_etb       '\027'
```

```
DEFINITION MODULE Architecture ;

(*
    Title      : Architecture
    Author     : Gaius Mulley
    System     : UNIX (gm2)
    Date       : Fri Jul 10 10:58:53 1998
    Last edit  : Fri Jul 10 10:58:53 1998
    Description: provides a relative speed indicator.
*)

EXPORT QUALIFIED RelativeSpeed, IsSimulation ;

CONST
   RelativeSpeed = 100000 ;
   IsSimulation  =  FALSE ;


END Architecture.
```



```
/* Header for module Architecture, generated by p2c */
#ifndef ArchitectureH
#define ArchitectureH


#ifdef ArchitectureG
# define vextern
#else
# define vextern extern
#endif


/*
    Title      : Architecture
    Author     : Gaius Mulley
    System     : UNIX (gm2)
    Date       : Fri Jul 10 10:58:53 1998
    Last edit  : Fri Jul 10 10:58:53 1998
    Description: provides a relative speed indicator.
*/


#define Architecture_RelativeSpeed  100000L

#define Architecture_IsSimulation  FALSE


#undef vextern

#endif /*ArchitectureH*/
```

```
DEFINITION MODULE BufferDevice ;

(*
   Author      : Gaius Mulley
   Version     : 1.0
   Date        : 12/2/86
   Last Update : 2/8/86
   Description : Exports a general Read and Write Buffer procedure. B
                 are dynamic.
*)

EXPORT QUALIFIED InitBuffer, KillBuffer, ReadBuffer, WriteBuffer,
                 Buffer ;

TYPE
   Buffer ;


(*
   InitBuffer - creates, and initializes, a buffer and returns it.
*)

PROCEDURE InitBuffer () : Buffer ;

(*
   KillBuffer - destroys the buffer and relinquishes all associated
                resources.
```



```
/* Header for module BufferDevice, generated by p2c */
#ifndef BufferDeviceH
#define BufferDeviceH


#ifdef BufferDeviceG
# define vextern
#else
# define vextern extern
#endif


/*
   Author      : Gaius Mulley
   Version     : 1.0
   Date        : 12/2/86
   Last Update : 2/8/86
   Description : Exports a general Read and Write Buffer procedure. B
                 are dynamic.
*/


#ifndef BufferDeviceG
/* opaque type declaration */
typedef void *BufferDevice_Buffer;

#endif
```

```
*)

PROCEDURE KillBuffer (VAR b: Buffer) ;


(*
   ReadBuffer - reads a character, ch, from the buffer, b.
*)

PROCEDURE ReadBuffer (b: Buffer ; VAR ch: CHAR) ;


(*
   WriteBuffer - places a character, ch, into buffer, b.
*)

PROCEDURE WriteBuffer (b: Buffer ; ch: CHAR) ;


END BufferDevice.
```



```
/*
   InitBuffer - creates, and initializes, a buffer and returns it.
*/


extern void *(BufferDevice_InitBuffer(void));

/*
   KillBuffer - destroys the buffer and relinquishes all associated
                resources.
*/

extern void BufferDevice_KillBuffer(void **b);


/*
   ReadBuffer - reads a character, ch, from the buffer, b.
*/

extern void BufferDevice_ReadBuffer(void *b, Char *ch);


/*
   WriteBuffer - places a character, ch, into buffer, b.
*/

extern void BufferDevice_WriteBuffer(void *b, Char ch);


#undef vextern

#endif /*BufferDeviceH*/

/* End. */
```

```
DEFINITION MODULE ClockDevice ;

(*
    Title      : ClockDevice
    Author     : Gaius Mulley
    System     : UNIX (gm2)
    Date       : Thu Aug 11 15:17:12 1994
    Last edit  : Thu Aug 11 15:17:12 1994
    Description: provides a simple interface to the 8253 timer chip
                 found on the IBM-PC. It only provides access to coun
                 in mode 0. Basically just enough for the Kernel.
*)

EXPORT QUALIFIED LoadClock, StartClock ;


(*
   LoadClock - returns the clock device current count.
*)

PROCEDURE LoadClock () : CARDINAL ;


(*
   StartClock - sets the Count into clock.
*)

PROCEDURE StartClock (Count: CARDINAL) ;
```



```
/* Header for module ClockDevice, generated by p2c */
#ifndef ClockDeviceH
#define ClockDeviceH


#ifdef ClockDeviceG
# define vextern
#else
# define vextern extern
#endif

/*
    Title      : ClockDevice
    Author     : Gaius Mulley
    System     : UNIX (gm2)
    Date       : Thu Aug 11 15:17:12 1994
    Last edit  : Thu Aug 11 15:17:12 1994
    Description: provides a simple interface to the 8253 timer chip
                 found on the IBM-PC. It only provides access to coun
                 in mode 0. Basically just enough for the Kernel.
*/



/*
   LoadClock - returns the clock device current count.
```

```
END ClockDevice.
```



```
*/

extern unsigned long ClockDevice_LoadClock(void);


/*
   StartClock - sets the Count into clock.
*/

extern void ClockDevice_StartClock(unsigned long Count);


#undef vextern

#endif /*ClockDeviceH*/

/* End. */
```

```
DEFINITION MODULE ColourText ;

(*
    Title      : ColourText
    Author     : Gaius Mulley
    System     : UNIX (gm2)
    Date       : Sun Jul 31 20:31:10 1994
    Last edit  : Sun Jul 31 20:31:10 1994
    Description: provides a simple interface to basic colour text.
*)

EXPORT QUALIFIED WriteCharacterAndAttribute ;


(*
   WriteCharacterAndAttribute - writes character, ch, to screen locat
                                x, y, with attibute colour, colour.
*)

PROCEDURE WriteCharacterAndAttribute (x, y: CARDINAL;
                                      ch: CHAR; colour: CARDINAL) ;



END ColourText.
```



```
/* Header for module ColourText, generated by p2c */
#ifndef ColourTextH
#define ColourTextH


#ifdef ColourTextG
# define vextern
#else
# define vextern extern
#endif

/*
    Title      : ColourText
    Author     : Gaius Mulley
    System     : UNIX (gm2)
    Date       : Sun Jul 31 20:31:10 1994
    Last edit  : Sun Jul 31 20:31:10 1994
    Description: provides a simple interface to basic colour text.
*/



/*
   WriteCharacterAndAttribute - writes character, ch, to screen locat
                                x, y, with attibute colour, colour.
*/
```

```
DEFINITION MODULE Colours ;


EXPORT QUALIFIED Black, Blue, Green, Cyan, Red, Magenta, Brown, Light
                 DarkGrey, LightBlue, LightGreen, LightCyan, LightRed
                 LightMagenta, Yellow, White ;


CONST
   Black        =  0 ;
   Blue         =  1 ;
   Green        =  2 ;
   Cyan         =  3 ;
   Red          =  4 ;
   Magenta      =  5 ;
   Brown        =  6 ;
   LightGrey    =  7 ;
   DarkGrey     =  8 ;
   LightBlue    =  9 ;
   LightGreen   = 10 ;
   LightCyan    = 11 ;
   LightRed     = 12 ;
   LightMagenta = 13 ;
   Yellow       = 14 ;
```



```
/* Header for module Colours, generated by p2c */
#ifndef ColoursH
#define ColoursH


#ifdef ColoursG
# define vextern
#else
# define vextern extern
#endif




#define Colours_Black   0
#define Colours_Blue    1
#define Colours_Green   2
#define Colours_Cyan    3
#define Colours_Red     4
#define Colours_Magenta  5
#define Colours_Brown   6
#define Colours_LightGrey  7
#define Colours_DarkGrey  8
#define Colours_LightBlue  9
#define Colours_LightGreen  10
#define Colours_LightCyan  11
```

```
   White        = 15 ;


END Colours.
```



```
#define Colours_LightRed  12
#define Colours_LightMagenta  13
#define Colours_Yellow  14
#define Colours_White   15


#undef vextern

#endif /*ColoursH*/

/* End. */
```

```
DEFINITION MODULE Debug ;

(*
    Title      : Debug
    Author     :
    System     : UNIX (gm2)
    Date       : Sat Aug 13 19:41:57 1994
    Last edit  : Sat Aug 13 19:41:57 1994
    Description: provides some simple debugging routines.
*)

EXPORT QUALIFIED Halt, DebugString ;


(*
   Halt - writes a message in the format:
          Module:Line:Message

          to the debugging device. (Scn.Write).
          It then terminates by looping forever.
*)

PROCEDURE Halt (Message: ARRAY OF CHAR;
                LineNo: CARDINAL;
                Module: ARRAY OF CHAR) ;


(*
```



```
/* Header for module Debug, generated by p2c */
#ifndef DebugH
#define DebugH


#ifdef DebugG
# define vextern
#else
# define vextern extern
#endif

/*
    Title      : Debug
    Author     :
    System     : UNIX (gm2)
    Date       : Sat Aug 13 19:41:57 1994
    Last edit  : Sat Aug 13 19:41:57 1994
    Description: provides some simple debugging routines.
*/



/*
   Halt - writes a message in the format:
          Module:Line:Message

          to the debugging device. (Scn.Write).
```

```
   DebugString - writes a string to the debugging device (Scn.Write).
                 It interprets \n as carriage return, linefeed.
*)

PROCEDURE DebugString (a: ARRAY OF CHAR) ;


END Debug.
```



```
          It then terminates by looping forever.
*/

extern void Debug_Halt(const long Message_LOW, const long Message_HIG
  const Char *Message, unsigned long LineNo, const long Module_LOW,
  const long Module_HIGH, const Char *Module);


/*
   DebugString - writes a string to the debugging device (Scn.Write).
                 It interprets \n as carriage return, linefeed.
*/

extern void Debug_DebugString(const long a_LOW, const long a_HIGH,
                  const Char *a);


#undef vextern

#endif /*DebugH*/

/* End. */
```

```
DEFINITION MODULE Descriptors ;

(*
    Title      : Descriptors
    Author     : Gaius Mulley
    System     : UNIX (gm2)
    Date       : Wed Aug  3 16:14:35 1994
    Last edit  : Wed Aug  3 16:14:35 1994
    Description: provides an interface to some assembly language
                 which sets up the 386/486 gdt and idt.
                 This does the minumum necessary to allow the 386/486
                 to access 1Gb
*)

FROM WordSizes IMPORT SHORTWORD ;

EXPORT QUALIFIED InterruptGate, SetupGDT, SetupIDT, IDT, AfterIDTGDT


TYPE
   InterruptGate = RECORD
                      Offset0To15 : SHORTWORD ;
                      Segment     : SHORTWORD ;
                      Flags       : SHORTWORD ;
                      Offset16To31: SHORTWORD ;
```



```
/* Header for module Descriptors, generated by p2c */
#ifndef DescriptorsH
#define DescriptorsH


/*
    Title      : Descriptors
    Author     : Gaius Mulley
    System     : UNIX (gm2)
    Date       : Wed Aug  3 16:14:35 1994
    Last edit  : Wed Aug  3 16:14:35 1994
    Description: provides an interface to some assembly language
                 which sets up the 386/486 gdt and idt.
                 This does the minumum necessary to allow the 386/486
                 to access 1Gb
*/

#ifndef WordSizesH
#include "WordSizes.h"
#endif


#ifdef DescriptorsG
# define vextern
#else
```

```
                END ;

VAR
   IDT: ARRAY [0..255] OF InterruptGate ;


(*
   SetupGDT - sets up a global descriptor for the whole of physical
              memory. It maps 16 Mb of physical memory.
              It does not touch any extended memory - you must be
              careful not to touch any physical memory that isn't
              really there.
*)

PROCEDURE SetupGDT ;


(*
   SetupIDT - not sure what this does...
*)

PROCEDURE SetupIDT ;


(*
   AfterIDTGDT - a label which exists at the end of our IDT and GDT t
*)

PROCEDURE AfterIDTGDT ;


END Descriptors.
```



```
# define vextern extern
#endif


typedef struct Descriptors_InterruptGate {
  WordSizes_SHORTWORD Offset0To15, Segment, Flags, Offset16To31;
} Descriptors_InterruptGate;


vextern Descriptors_InterruptGate Descriptors_IDT[256];


/*
   SetupGDT - sets up a global descriptor for the whole of physical
              memory. It maps 16 Mb of physical memory.
              It does not touch any extended memory - you must be
              careful not to touch any physical memory that isn't
              really there.
*/

extern void Descriptors_SetupGDT(void);


/*
   SetupIDT - not sure what this does...
*/

extern void Descriptors_SetupIDT(void);


/*
   AfterIDTGDT - a label which exists at the end of our IDT and GDT t
```

```
*/

extern void Descriptors_AfterIDTGDT(void);


#undef vextern

#endif /*DescriptorsH*/

/* End. */
```

```
DEFINITION MODULE DeviceConfiguration ;

(*
   Title      : DeviceConfiguration
   Author     : Gaius Mulley
   Date       : 28/2/88
   LastEdit   : 7/9/94
   System     : RTS gm2
   Description: Provides a global module for device characteristics
                and device names.
*)

EXPORT QUALIFIED Parity, GetIrqNo ;

TYPE
   Parity = (None, Odd, Even) ;


(*
   GetIrqNo - returns the irq number for a UART.
              This function returns 256 if UART has no IRQ
              else IRQ level (2-7)
*)

PROCEDURE GetIrqNo (BaseAddress: CARDINAL) : CARDINAL ;


END DeviceConfiguration.
```



```
/* Header for module DeviceConfiguration, generated by p2c */
#ifndef DeviceConfigurationH
#define DeviceConfigurationH


#ifdef DeviceConfigurationG
# define vextern
#else
# define vextern extern
#endif


/*
   Title      : DeviceConfiguration
   Author     : Gaius Mulley
   Date       : 28/2/88
   LastEdit   : 7/9/94
   System     : RTS gm2
   Description: Provides a global module for device characteristics
                and device names.
*/


typedef enum {
  DeviceConfiguration_None, DeviceConfiguration_Odd, DeviceConfigurat
} DeviceConfiguration_Parity;
```

```
/*
   GetIrqNo - returns the irq number for a UART.
              This function returns 256 if UART has no IRQ
              else IRQ level (2-7)
*/


extern unsigned long DeviceConfiguration_GetIrqNo(unsigned long BaseA


#undef vextern

#endif /*DeviceConfigurationH*/

/* End. */
```

```
DEFINITION MODULE Dump ;

(*
    Title      : Dump
    Author     : Gaius Mulley
    System     : UNIX (gm2)
    Date       : Thu Aug  4 16:39:08 1994
    Last edit  : Thu Aug  4 16:39:08 1994
    Description: provides a crude hex dump of memory.
*)

FROM SYSTEM IMPORT ADDRESS ;
EXPORT QUALIFIED DumpDown, DumpUp ;


(*
   DumpDown - displays Length bytes in WORDs starting at, Top,
              and working down.
*)

PROCEDURE DumpDown (Top: ADDRESS; Length: CARDINAL) ;


(*
   DumpUp - displays Length bytes in WORDs starting at, Bot,
            and working up.
*)
```



```
/* Header for module Dump, generated by p2c */
#ifndef DumpH
#define DumpH


#ifndef SYSTEMH
#include "SYSTEM.h"
#endif


#ifdef DumpG
# define vextern
#else
# define vextern extern
#endif

/*
    Title      : Dump
    Author     : Gaius Mulley
    System     : UNIX (gm2)
    Date       : Thu Aug  4 16:39:08 1994
    Last edit  : Thu Aug  4 16:39:08 1994
    Description: provides a crude hex dump of memory.
*/


/*
```

```
PROCEDURE DumpUp (Bot: ADDRESS; Length: CARDINAL) ;


END Dump.
```



```
   DumpDown - displays Length bytes in WORDs starting at, Top,
              and working down.
*/

extern void Dump_DumpDown(void *Top, unsigned long Length);


/*
   DumpUp - displays Length bytes in WORDs starting at, Bot,
            and working up.
*/

extern void Dump_DumpUp(void *Bot, unsigned long Length);


#undef vextern

#endif /*DumpH*/

/* End. */
```

```
DEFINITION MODULE Exceptions ;

(*
    Title      : Exceptions
    Author     : Gaius Mulley
    System     : RTS (gm2)
    Date       : Thu Aug 11 08:19:21 1994
    Last edit  : Thu Aug 11 08:19:21 1994
    Description: provides a simple? set of 386/486 exception handlers
*)

EXPORT QUALIFIED Init ;


(*
   Init - install a set of 386/486 exception handlers.
*)

PROCEDURE Init ;


END Exceptions.
```



```
/* Header for module Exceptions, generated by p2c */
#ifndef ExceptionsH
#define ExceptionsH


#ifdef ExceptionsG
# define vextern
#else
# define vextern extern
#endif

/*
    Title      : Exceptions
    Author     : Gaius Mulley
    System     : RTS (gm2)
    Date       : Thu Aug 11 08:19:21 1994
    Last edit  : Thu Aug 11 08:19:21 1994
    Description: provides a simple? set of 386/486 exception handlers
*/



/*
   Init - install a set of 386/486 exception handlers.
*/

extern void Exceptions_Init(void);
```

```
DEFINITION MODULE Executive ;

(*
    Title      : Executive
    Author     : Gaius Mulley
    System     : UNIX (gm2)
    Date       : Thu Aug 18 10:34:58 1994
    Last edit  : Thu Aug 18 10:34:58 1994
    Description: provides a simple multitasking executive.
*)

EXPORT QUALIFIED SEMAPHORE, DESCRIPTOR,
                 InitProcess, Resume, Suspend, InitSemaphore, Wait, S
                 WaitForIO, Ps, GetCurrentProcess, RotateRunQueue,
                 ProcessName, DebugProcess ;

TYPE
   SEMAPHORE ;         (* defines dijkstra's semaphores *)
   DESCRIPTOR ;        (* handle onto a process         *)


(*
   InitProcess - initializes a process which is held in the suspended
                 state. When the process is resumed it will start exe
                 procedure, p. The process has a maximum stack size o
```



```
#ifndef ExecutiveH
#define ExecutiveH


/*
    Title      : Executive
    Author     : Gaius Mulley
    System     : UNIX (gm2)
    Date       : Thu Aug 18 10:34:58 1994
    Last edit  : Thu Aug 18 10:34:58 1994
    Description: provides a simple multitasking executive.
*/


#ifndef ExecutiveG
typedef void *Executive_SEMAPHORE;  /* opaque type declaration */
                                    /* defines dijkstra's semaphores
#endif

#ifndef ExecutiveG
typedef void *Executive_DESCRIPTOR; /* opaque type declaration */
                                    /* handle onto a process
#endif


/*
   InitProcess - initializes a process which is held in the suspended
```

```
                StackSize, bytes and its textual name is, Name.
                The StackSize should be at least 5000 bytes.
*)

PROCEDURE InitProcess (p: PROC; StackSize: CARDINAL;
                       Name: ARRAY OF CHAR) : DESCRIPTOR ;


(*
   Resume - resumes a suspended process. If all is successful then th
            is returned. If it fails then NIL is returned.
*)

PROCEDURE Resume (d: DESCRIPTOR) : DESCRIPTOR ;


(*
   Suspend - suspend the calling process.
             The process can only continue running if another process
             Resumes it.
*)

PROCEDURE Suspend ;


(*
   InitSemaphore - creates a semaphore whose initial value is, v, and
                   whose name is, Name.
*)
```



```
                state. When the process is resumed it will start exe
                procedure, p. The process has a maximum stack size o
                StackSize, bytes and its textual name is, Name.
                The StackSize should be at least 5000 bytes.
*/


extern void *(Executive_InitProcess(void (*p)(void), unsigned long St
        const long Name_LOW, const long Name_HIGH, const Char *Name))


/*
   Resume - resumes a suspended process. If all is successful then th
            is returned. If it fails then NIL is returned.
*/

extern void *(Executive_Resume(void *d));


/*
   Suspend - suspend the calling process.
             The process can only continue running if another process
             Resumes it.
*/

extern void Executive_Suspend(void);


/*
   InitSemaphore - creates a semaphore whose initial value is, v, and
```

```
PROCEDURE InitSemaphore (v: CARDINAL; Name: ARRAY OF CHAR) : SEMAPHOR

(*
   Wait - performs dijkstra's P operation on a semaphore.
          A process which calls this procedure will
          wait until the value of the semaphore is > 0
          and then it will decrement this value.
*)

PROCEDURE Wait (s: SEMAPHORE) ;


(*
   Signal - performs dijkstra's V operation on a semaphore.
            A process which calls the procedure will increment
            the semaphores value.
*)

PROCEDURE Signal (s: SEMAPHORE) ;


(*
   WaitForIO - waits for an interrupt to occur on vector, VectorNo.
*)

PROCEDURE WaitForIO (VectorNo: CARDINAL) ;


(*
   Ps - displays a process list together with process status.
```



```
                  whose name is, Name.
*/

extern void *(Executive_InitSemaphore(unsigned long v, const long Nam
        const long Name_HIGH, const Char *Name));


/*
   Wait - performs dijkstra's P operation on a semaphore.
          A process which calls this procedure will
          wait until the value of the semaphore is > 0
          and then it will decrement this value.
*/

extern void Executive_Wait(void *s);


/*
   Signal - performs dijkstra's V operation on a semaphore.
            A process which calls the procedure will increment
            the semaphores value.
*/

extern void Executive_Signal(void *s);


/*
   WaitForIO - waits for an interrupt to occur on vector, VectorNo.
*/
```

```
*)

PROCEDURE Ps ;


(*
   GetCurrentProcess - returns the descriptor of the current running
                       process.
*)

PROCEDURE GetCurrentProcess () : DESCRIPTOR ;


(*
   RotateRunQueue - rotates the process run queue.
                    It does not call the scheduler.
*)

PROCEDURE RotateRunQueue ;


(*
   ProcessName - displays the name of process, d, through
                 DebugString.
*)

PROCEDURE ProcessName (d: DESCRIPTOR) ;


(*
   DebugProcess - gdb debug handle to enable users to debug deadlocke
                  semaphore processes.
```



```
extern void Executive_WaitForIO(unsigned long VectorNo);


/*
   Ps - displays a process list together with process status.
*/

extern void Executive_Ps(void);


/*
   GetCurrentProcess - returns the descriptor of the current running
                       process.
*/

extern void *(Executive_GetCurrentProcess(void));


/*
   RotateRunQueue - rotates the process run queue.
                    It does not call the scheduler.
*/

extern void Executive_RotateRunQueue(void);


/*
   ProcessName - displays the name of process, d, through
                 DebugString.
*/

extern void Executive_ProcessName(void *d);
```

```
*)

PROCEDURE DebugProcess (d: DESCRIPTOR) ;


END Executive.
```



```
/*
   DebugProcess - gdb debug handle to enable users to debug deadlocke
                  semaphore processes.
*/

extern void Executive_DebugProcess(void *d);


#endif /*ExecutiveH*/
```

```
DEFINITION MODULE IO ;

(*
      Author     : Gaius Mulley
      Title      : IO
      Version    : 1.0
      Date       : 12/2/86
      Last Edit  : 12/2/86
      Description: Provides a mechanism whereby all processes can use
                   same input/output libraries but communicate to dif
                   devices.
*)

FROM SYSTEM IMPORT ADDRESS ;
FROM StdIO IMPORT ProcWrite, ProcRead ;

EXPORT QUALIFIED NewStream, DisposeStream, ConnectStream, DisConnectS
                 SwapStream, InitStream, DupStream,
                 STREAM ;

TYPE
   STREAM = POINTER TO Stream ;
   Stream = RECORD
               inchar : ProcRead ;
```



```
/* Header for module IO, generated by p2c */
#ifndef IOH
#define IOH


#ifndef SYSTEMH
#include "SYSTEM.h"
#endif

/*
      Author     : Gaius Mulley
      Title      : IO
      Version    : 1.0
      Date       : 12/2/86
      Last Edit  : 12/2/86
      Description: Provides a mechanism whereby all processes can use
                   same input/output libraries but communicate to dif
                   devices.
*/

#ifndef StdIOH
#include "StdIO.h"
#endif


#ifdef IOG
```

```
            outchar: ProcWrite ;
            errchar: ProcWrite ;
            devin  : ADDRESS ;
            devout : ADDRESS ;
            deverr : ADDRESS ;
         END ;


(*
   NewStream - creates a new stream, s.
*)

PROCEDURE NewStream (VAR s: STREAM) ;

(*
   DisposeStream - disposes of an unused stream, s.
*)

PROCEDURE DisposeStream (VAR s: STREAM) ;

(*
   ConnectStream - sets stream s1 to stream s2.
*)

PROCEDURE ConnectStream (VAR s1: STREAM ; s2: STREAM) ;

(*
   DisConnectStream - sets stream s2 to stream s1.
```



```
# define vextern
#else
# define vextern extern
#endif


typedef struct IO_Stream {
  void (*inchar)(Char *);
  void (*outchar)(Char), (*errchar)(Char);
  void *devin, *devout, *deverr;
} IO_Stream;


/*
   NewStream - creates a new stream, s.
*/


extern void IO_NewStream(IO_Stream **s);

/*
   DisposeStream - disposes of an unused stream, s.
*/

extern void IO_DisposeStream(IO_Stream **s);

/*
   ConnectStream - sets stream s1 to stream s2.
*/
```

```
*)

PROCEDURE DisConnectStream (s1: STREAM ; VAR s2: STREAM) ;

(*
   Swaps the streams s1 and s2.
*)

PROCEDURE SwapStream (VAR s1, s2: STREAM) ;


(*
   DupStream - duplicates the stream, s1, a copy is made in s2.
*)

PROCEDURE DupStream (s1: STREAM; s2: STREAM) ;


(*
   InitStream - initialises a stream, s, to contain a read procedure
                r, a write procedure, w, and an error procedure e.
                The Addresses ra, wa, ea, at the moment have little
                significance - one day they will be used to contain
                data blocks for block devices.
*)

PROCEDURE InitStream (s: STREAM ;
                      r: ProcRead ; ra: ADDRESS ;
                      w: ProcWrite ; wa: ADDRESS ;
```



```
extern void IO_ConnectStream(IO_Stream **s1, IO_Stream *s2);

/*
   DisConnectStream - sets stream s2 to stream s1.
*/

extern void IO_DisConnectStream(IO_Stream *s1, IO_Stream **s2);

/*
   Swaps the streams s1 and s2.
*/

extern void IO_SwapStream(IO_Stream **s1, IO_Stream **s2);


/*
   DupStream - duplicates the stream, s1, a copy is made in s2.
*/

extern void IO_DupStream(IO_Stream *s1, IO_Stream *s2);


/*
   InitStream - initialises a stream, s, to contain a read procedure
                r, a write procedure, w, and an error procedure e.
                The Addresses ra, wa, ea, at the moment have little
                significance - one day they will be used to contain
                data blocks for block devices.
*/
```

```
        e: ProcWrite ; ea: ADDRESS) ;
END IO.
```



```
extern void IO_InitStream(IO_Stream *s, void (*r)(Char *), void *ra,
                          void (*w)(Char), void *wa, void (*e)(Char),
                          void *ea);


#undef vextern

#endif /*IOH*/

/* End. */
```

```
DEFINITION MODULE IRQ ;

(*
    Title      : IRQ
    Author     : Gaius Mulley
    System     : RTS (gm2)
    Date       : Tue Aug  9 12:11:54 1994
    Last edit  : Tue Aug  9 12:11:54 1994
    Description: provides a simple interface to the IRQ hardware.
*)

EXPORT QUALIFIED ProcIRQ, InitIRQ, KillIRQ, EnableIRQ, DisableIRQ, In


CONST
   MaxIRQ = 15 ;

TYPE
   ProcIRQ = PROCEDURE (CARDINAL) ;


(*
   InitIRQ - initialises irq, IrqNo, to call PROCEDURE, p,
             when this interrupt occurs.
*)

PROCEDURE InitIRQ (IrqNo: CARDINAL; p: ProcIRQ) ;


(*
```



```
/* Header for module IRQ, generated by p2c */
#ifndef IRQH
#define IRQH


#ifdef IRQG
# define vextern
#else
# define vextern extern
#endif


/*
    Title      : IRQ
    Author     : Gaius Mulley
    System     : RTS (gm2)
    Date       : Tue Aug  9 12:11:54 1994
    Last edit  : Tue Aug  9 12:11:54 1994
    Description: provides a simple interface to the IRQ hardware.
*/



#define IRQ_MaxIRQ      15


typedef void (*IRQ_ProcIRQ)(unsigned long);



/*
   InitIRQ - initialises irq, IrqNo, to call PROCEDURE, p,
```

```
   KillIRQ - removes an IRQ handler.
*)

PROCEDURE KillIRQ (IrqNo: CARDINAL) ;


(*
   EnableIRQ - enable irq, IrqNo.
*)

PROCEDURE EnableIRQ (IrqNo: CARDINAL) ;


(*
   DisableIRQ - disable irq, IrqNo.
*)

PROCEDURE DisableIRQ (IrqNo: CARDINAL) ;


(*
   Init - initializes the module data structures and assigns default
          irq handlers.
*)

PROCEDURE Init ;


END IRQ.
```



```
             when this interrupt occurs.
*/


extern void IRQ_InitIRQ(unsigned long IrqNo, void (*p)(unsigned long)


/*
   KillIRQ - removes an IRQ handler.
*/

extern void IRQ_KillIRQ(unsigned long IrqNo);


/*
   EnableIRQ - enable irq, IrqNo.
*/

extern void IRQ_EnableIRQ(unsigned long IrqNo);


/*
   DisableIRQ - disable irq, IrqNo.
*/

extern void IRQ_DisableIRQ(unsigned long IrqNo);


/*
   Init - initializes the module data structures and assigns default
          irq handlers.
*/

extern void IRQ_Init(void);
```

```
DEFINITION MODULE InterruptVector ;

(*
    Title      : InterruptVector
    Author     : Gaius Mulley
    System     : RTS (gm2)
    Date       : Mon Aug  8 19:14:59 1994
    Last edit  : Mon Aug  8 19:14:59 1994
    Description: provides a simple interface to interrupt vectors.
*)

FROM SYSTEM IMPORT ADDRESS ;

EXPORT QUALIFIED ProcISR,
                 InitInterruptVector, KillInterruptVector, Init,
                 InstallIsr ;

TYPE
   ProcISR = PROCEDURE (CARDINAL) ;


(*
   InitInterruptVector - initializes interrupt, VectorNo, to call
                         PROCEDURE, p, when an interrupt occurs.
*)

PROCEDURE InitInterruptVector (VectorNo: CARDINAL; p: ProcISR) ;
```



```
/* Header for module InterruptVector, generated by p2c */
#ifndef InterruptVectorH
#define InterruptVectorH


#ifndef SYSTEMH
#include "SYSTEM.h"
#endif


#ifdef InterruptVectorG
# define vextern
#else
# define vextern extern
#endif


/*
    Title      : InterruptVector
    Author     : Gaius Mulley
    System     : RTS (gm2)
    Date       : Mon Aug  8 19:14:59 1994
    Last edit  : Mon Aug  8 19:14:59 1994
    Description: provides a simple interface to interrupt vectors.
*/


typedef void (*InterruptVector_ProcISR)(unsigned long);
```

```
(*
   KillInterruptVector - removes interrupt, VectorNo.
*)

PROCEDURE KillInterruptVector (VectorNo: CARDINAL) ;


(*
   InstallIsr - installs an interrupt service routine at interrupt ve
                VectorNo, which will call address, a.
                (A very low level routine which is only exported so t
                 debugger stub can get access to the breakpoint et al
*)

PROCEDURE InstallIsr (VectorNo: CARDINAL; a: ADDRESS) ;


(*
   Init - sets up IsrTemplate and then initializes all interrupt vect
*)

PROCEDURE Init ;


END InterruptVector.
```



```
/*
   InitInterruptVector - initializes interrupt, VectorNo, to call
                         PROCEDURE, p, when an interrupt occurs.
*/


extern void InterruptVector_InitInterruptVector(unsigned long VectorN
                         void (*p)(unsigned long));


/*
   KillInterruptVector - removes interrupt, VectorNo.
*/

extern void InterruptVector_KillInterruptVector(unsigned long VectorN


/*
   InstallIsr - installs an interrupt service routine at interrupt ve
                VectorNo, which will call address, a.
                (A very low level routine which is only exported so t
                 debugger stub can get access to the breakpoint et al
*/

extern void InterruptVector_InstallIsr(unsigned long VectorNo, void *


/*
   Init - sets up IsrTemplate and then initializes all interrupt vect
*/
```

```
DEFINITION MODULE KeyBoardConvert ;


FROM StdIO IMPORT ProcWrite ;

EXPORT QUALIFIED ConsoleSwitchProc, ScanToASCII, SetFunctionString ;


TYPE
   ConsoleSwitchProc = PROCEDURE (CARDINAL) ;


PROCEDURE ScanToASCII (ReadDeliver: ProcWrite ; ch: CHAR; ConsoleSwit


(*
   SetFunctionString - sets a function key to deliver a string, a,
                       when pressed.
*)

PROCEDURE SetFunctionString (Function: CARDINAL; a: ARRAY OF CHAR) ;


END KeyBoardConvert.
```



```
/* Header for module KeyBoardConvert, generated by p2c */
#ifndef KeyBoardConvertH
#define KeyBoardConvertH


#ifndef StdIOH
#include "StdIO.h"
#endif


#ifdef KeyBoardConvertG
# define vextern
#else
# define vextern extern
#endif



typedef void (*KeyBoardConvert_ConsoleSwitchProc)(unsigned long);



extern void KeyBoardConvert_ScanToASCII(void (*ReadDeliver)(Char), Ch
                     void (*ConsoleSwitch)(unsigned long));


/*
   SetFunctionString - sets a function key to deliver a string, a,
                       when pressed.
*/

extern void KeyBoardConvert_SetFunctionString(unsigned long Function,
  const long a_LOW, const long a_HIGH, const Char *a);
```

```
DEFINITION MODULE KeyBoardLEDs ;

(*
    Title      : KeyBoardLEDs
    Author     : Gaius Mulley
    System     : UNIX (gm2)
    Date       : Fri Aug 12 17:05:58 1994
    Last edit  : Fri Aug 12 17:05:58 1994
    Description: provides a simple packages to manipulate the keyboar
                 LEDs.

                 This module remains for compatability reasons only.
                 Please alter code to use TTIO in the future.
*)

EXPORT QUALIFIED SwitchLeds,
                 SwitchScroll, SwitchNum, SwitchCaps ;


(*
   SwitchLeds - switch the keyboard LEDs to the state defined
                by the BOOLEAN variables. TRUE = ON.
*)

PROCEDURE SwitchLeds (NumLock, CapsLock, ScrollLock: BOOLEAN) ;


(*
```



```
/* Header for module KeyBoardLEDs, generated by p2c */
#ifndef KeyBoardLEDsH
#define KeyBoardLEDsH


#ifdef KeyBoardLEDsG
# define vextern
#else
# define vextern extern
#endif

/*
    Title      : KeyBoardLEDs
    Author     : Gaius Mulley
    System     : UNIX (gm2)
    Date       : Fri Aug 12 17:05:58 1994
    Last edit  : Fri Aug 12 17:05:58 1994
    Description: provides a simple packages to manipulate the keyboar
                 LEDs.

                 This module remains for compatability reasons only.
                 Please alter code to use TTIO in the future.
*/


/*
```

```
   SwitchScroll - switchs the scroll LED on or off.
*)

PROCEDURE SwitchScroll (Scroll: BOOLEAN) ;


(*
   SwitchNum - switches the Num LED on or off.
*)

PROCEDURE SwitchNum (Num: BOOLEAN) ;


(*
   SwitchCaps - switches the Caps LED on or off.
*)

PROCEDURE SwitchCaps (Caps: BOOLEAN) ;


END KeyBoardLEDs.
```



```
   SwitchLeds - switch the keyboard LEDs to the state defined
                by the BOOLEAN variables. TRUE = ON.
*/

extern void KeyBoardLEDs_SwitchLeds(BOOLEAN NumLock, BOOLEAN CapsLock
                    BOOLEAN ScrollLock);


/*
   SwitchScroll - switchs the scroll LED on or off.
*/

extern void KeyBoardLEDs_SwitchScroll(BOOLEAN Scroll);


/*
   SwitchNum - switches the Num LED on or off.
*/

extern void KeyBoardLEDs_SwitchNum(BOOLEAN Num);


/*
   SwitchCaps - switches the Caps LED on or off.
*/

extern void KeyBoardLEDs_SwitchCaps(BOOLEAN Caps);


#undef vextern

#endif /*KeyBoardLEDsH*/

/* End. */
```

```
DEFINITION MODULE M2RTS ;

(*
   Author     : Gaius Mulley
   Title      : M2RTS
   Date       : Wed Jun 20 15:21:04 BST 1990
   Description: Implements the Run Time System facilities of Modula-2
   Last update: Wed Jun 20 15:21:39 BST 1990
*)

EXPORT QUALIFIED HALT, InstallTerminationProcedure, Terminate,
                 ExitOnHalt,
                 SubrangeAssignmentError, ArraySubscriptError ;



(*
   HALT - terminate the current program calling creating a core dump.
          The procedure Terminate is called before the core dump is
          created.
*)

PROCEDURE HALT ;


(*
   SubrangeAssignmentError - part of the runtime checking, called if
                             subrange variable is just about to be as
```



```
/* Header for module M2RTS, generated by p2c */
#ifndef M2RTSH
#define M2RTSH


#ifdef M2RTSG
# define vextern
#else
# define vextern extern
#endif

/*
   Author     : Gaius Mulley
   Title      : M2RTS
   Date       : Wed Jun 20 15:21:04 BST 1990
   Description: Implements the Run Time System facilities of Modula-2
   Last update: Wed Jun 20 15:21:39 BST 1990
*/



/*
   HALT - terminate the current program calling creating a core dump.
          The procedure Terminate is called before the core dump is
          created.
*/
```

```
*)

PROCEDURE SubrangeAssignmentError (file: ARRAY OF CHAR; line: CARDINA


(*
   ArraySubscriptError -  part of the runtime checking, called if an
                          array indice is out of range.
*)

PROCEDURE ArraySubscriptError (file: ARRAY OF CHAR; line: CARDINAL) ;


(*
   ExitOnHalt - if HALT is executed then call exit with the exit code
*)

PROCEDURE ExitOnHalt (e: INTEGER) ;


(*
   InstallTerminationProcedure - installs a procedure, p, which will
                                 be called when the procedure Termina
                                 is ionvoked.
*)

PROCEDURE InstallTerminationProcedure (p: PROC) ;


(*
   Terminate - calls each installed termination procedure in turn.
*)
```



```
extern void M2RTS_HALT__(void);


/*
   SubrangeAssignmentError - part of the runtime checking, called if
                             subrange variable is just about to be as
*/

extern void M2RTS_SubrangeAssignmentError__(const long file_LOW,
  const long file_HIGH, const Char *file, unsigned long line);


/*
   ArraySubscriptError -  part of the runtime checking, called if an
                          array indice is out of range.
*/

extern void M2RTS_ArraySubscriptError__(const long file_LOW,
  const long file_HIGH, const Char *file, unsigned long line);


/*
   ExitOnHalt - if HALT is executed then call exit with the exit code
*/

extern void M2RTS_ExitOnHalt__(long e);


/*
   InstallTerminationProcedure - installs a procedure, p, which will
                                 be called when the procedure Termina
```

```
PROCEDURE Terminate ;


END M2RTS.
```



```
                                       is ionvoked.
*/

extern void M2RTS_InstallTerminationProcedure__(void (*p)(void));


/*
   Terminate - calls each installed termination procedure in turn.
*/

extern void M2RTS_Terminate__(void);


#undef vextern

#endif /*M2RTSH*/

/* End. */
```

```
DEFINITION MODULE MATH ;

(*
    Title      : MATH
    Author     : Gaius Mulley
    System     : UNIX (gm2)
    Date       : Thu Jan  5 12:34:33 1995
    Last edit  : Thu Jan  5 12:34:33 1995
    Description: pseudo module for the Modula-2 compiler.
                 It defines and exports all FPU built in math operati
*)

EXPORT QUALIFIED pi,
                 sin, cos, tan, atan, sqrt ;

(*
   CONST
      pi = 3.1415927 ; etc

   PROCEDURE sin (r: REAL/LONGREAL) : REAL/LONGREAL ;
   PROCEDURE cos (r: REAL/LONGREAL) : REAL/LONGREAL ;
   PROCEDURE tan (r: REAL/LONGREAL) : REAL/LONGREAL ;
   PROCEDURE atan (r: REAL/LONGREAL) : REAL/LONGREAL ;
   PROCEDURE sqrt (r: REAL/LONGREAL) : REAL/LONGREAL ;
```



```
/* Header for module MATH, generated by p2c */
#ifndef MATHH
#define MATHH


#ifdef MATHG
# define vextern
#else
# define vextern extern
#endif

/*
    Title      : MATH
    Author     : Gaius Mulley
    System     : UNIX (gm2)
    Date       : Thu Jan  5 12:34:33 1995
    Last edit  : Thu Jan  5 12:34:33 1995
    Description: pseudo module for the Modula-2 compiler.
                 It defines and exports all FPU built in math operati
*/


/*
   CONST
      pi = 3.1415927 ; etc
```

```
*)


END MATH.
```



```
   PROCEDURE sin (r: REAL/LONGREAL) : REAL/LONGREAL ;
   PROCEDURE cos (r: REAL/LONGREAL) : REAL/LONGREAL ;
   PROCEDURE tan (r: REAL/LONGREAL) : REAL/LONGREAL ;
   PROCEDURE atan (r: REAL/LONGREAL) : REAL/LONGREAL ;
   PROCEDURE sqrt (r: REAL/LONGREAL) : REAL/LONGREAL ;
*/


#undef vextern

#endif /*MATHH*/

/* End. */
```

```
DEFINITION MODULE MemRegion ;

(*
    Title      : MemRegion
    Author     : Gaius Mulley
    System     : UNIX (gm2)
    Date       : Thu Aug  7 08:45:29 1997
    Last edit  : Thu Aug  7 08:45:29 1997
    Description: provides a simple interface to the linker created
                 symbols which define the regions of memory.
*)

FROM SYSTEM IMPORT ADDRESS ;
EXPORT QUALIFIED EndOfOS ;


(*
   EndOfOS - returns an address which is the first byte after the OS.
*)

PROCEDURE EndOfOS () : ADDRESS ;


END MemRegion.
```



```
/* Header for module MemRegion, generated by p2c */
#ifndef MemRegionH
#define MemRegionH


#ifndef SYSTEMH
#include "SYSTEM.h"
#endif


#ifdef MemRegionG
# define vextern
#else
# define vextern extern
#endif

/*
    Title      : MemRegion
    Author     : Gaius Mulley
    System     : UNIX (gm2)
    Date       : Thu Aug  7 08:45:29 1997
    Last edit  : Thu Aug  7 08:45:29 1997
    Description: provides a simple interface to the linker created
                 symbols which define the regions of memory.
*/
```

```
DEFINITION MODULE MemUtils ;

(*
    Title      : MemUtils
    Author     : Gaius Mulley
    System     : UNIX (gm2)
    Date       : Tue Oct  1 13:59:56 1996
    Last edit  : Tue Oct  1 13:59:56 1996
    Description: provides some basic memory utilities.
*)

FROM SYSTEM IMPORT ADDRESS ;
EXPORT QUALIFIED MemCopy, MemZero ;


(*
   MemCopy - copys a region of memory to the required destination.
*)

PROCEDURE MemCopy (from: ADDRESS; length: CARDINAL; to: ADDRESS) ;


(*
   MemZero - sets a region of memory: a..a+length to zero.
*)

PROCEDURE MemZero (a: ADDRESS; length: CARDINAL) ;


END MemUtils.
```



```
/* Header for module MemUtils, generated by p2c */
#ifndef MemUtilsH
#define MemUtilsH


#ifndef SYSTEMH
#include "SYSTEM.h"
#endif


#ifdef MemUtilsG
# define vextern
#else
# define vextern extern
#endif

/*
    Title      : MemUtils
    Author     : Gaius Mulley
    System     : UNIX (gm2)
    Date       : Tue Oct  1 13:59:56 1996
    Last edit  : Tue Oct  1 13:59:56 1996
    Description: provides some basic memory utilities.
*/


/*
```

```
   MemCopy - copys a region of memory to the required destination.
*/

extern void MemUtils_MemCopy(void *from, unsigned long length, void *

/*
   MemZero - sets a region of memory: a..a+length to zero.
*/

extern void MemUtils_MemZero(void *a, unsigned long length);


#undef vextern

#endif /*MemUtilsH*/

/* End. */
```

```
DEFINITION MODULE MetaStore ;

(*
    Title      : MetaStore
    Author     : Gaius Mulley
    System     : UNIX (gm2)
    Date       : Mon Apr 14 09:51:22 1997
    Last edit  : Mon Apr 14 09:51:23 1997
    Description: provides the mechanisms for handling dynamic memory.
*)

FROM SYSTEM IMPORT ADDRESS ;

EXPORT QUALIFIED HEAP, AllocateMethod,
                 InitHeap, BestFit, FirstFit, Allocate, Deallocate, R
                 VerifyHeap, GetTotalSize, GetUsedSize, GetFreeSize,
                 GetNoOfFreeFragments, GetNoOfUsedFragments ;


TYPE
   HEAP ;
   AllocateMethod = PROCEDURE (HEAP, CARDINAL) : ADDRESS ;


(*
   InitHeap - creates a heap and returns a handle onto this heap.
              The start and size of the heap is defined by, a, and, s
```



```
/* Header for module MetaStore, generated by p2c */
#ifndef MetaStoreH
#define MetaStoreH


#ifndef SYSTEMH
#include "SYSTEM.h"
#endif


#ifdef MetaStoreG
# define vextern
#else
# define vextern extern
#endif


/*
    Title      : MetaStore
    Author     : Gaius Mulley
    System     : UNIX (gm2)
    Date       : Mon Apr 14 09:51:22 1997
    Last edit  : Mon Apr 14 09:51:23 1997
    Description: provides the mechanisms for handling dynamic memory.
*/



#ifndef MetaStoreG
```

```
              AllocP is the allocation method chosen.
*)

PROCEDURE InitHeap (a: ADDRESS; size: CARDINAL; AllocP: AllocateMetho


(*
   BestFit - chooses the best fit fragment in a heap.
*)

PROCEDURE BestFit (h: HEAP; size: CARDINAL) : ADDRESS ;


(*
   FirstFit - chooses the first fit fragment in a heap.
*)

PROCEDURE FirstFit (h: HEAP; size: CARDINAL) : ADDRESS ;


(*
   WorstFit - nah, nobody wants this do they?
*)

(* PROCEDURE WorstFit (h: HEAP; size: CARDINAL) : ADDRESS ; *)


(*
   Allocate - allocates, n, bytes of store from the heap.
*)

PROCEDURE Allocate (h: HEAP; n: CARDINAL) : ADDRESS ;


(*
```



```
/* opaque type declaration */
typedef void *MetaStore_HEAP;

#endif
typedef  (*MetaStore_AllocateMethod)(, unsigned long);



/*
   InitHeap - creates a heap and returns a handle onto this heap.
              The start and size of the heap is defined by, a, and, s
              AllocP is the allocation method chosen.
*/


extern void *(MetaStore_InitHeap(void *a, unsigned long size,
                  void *((*AllocP)(, unsigned long))));


/*
   BestFit - chooses the best fit fragment in a heap.
*/

extern void *(MetaStore_BestFit(void *h, unsigned long size));


/*
   FirstFit - chooses the first fit fragment in a heap.
*/

extern void *(MetaStore_FirstFit(void *h, unsigned long size));


/*
```

```
   Deallocate - deallocates, n, bytes of store.
*)

PROCEDURE Deallocate (h: HEAP; a: ADDRESS; size: CARDINAL) ;


(*
   ReAllocate - attempts to reallocate one chunk as two. It assumes t
                address, a, has already been allocated with fromsize
                Upon return there are two fragments allocated:

                a with tosize bytes
                b with bsize  bytes

                Note that tosize + bsize may not be equal to fromsize
                     as there is a system overhead.
*)

PROCEDURE Reallocate (h: HEAP;
                      a: ADDRESS; fromsize, tosize: CARDINAL;
                      VAR b: ADDRESS; VAR bsize: CARDINAL) ;


(*
   GetLargestFree - returns the largest free space fragment size in b
                    in a heap. NOTE it does not allocate this fragmen
*)

PROCEDURE GetLargestFree (h: HEAP) : CARDINAL ;
```



```
   WorstFit - nah, nobody wants this do they?
*/

/* PROCEDURE WorstFit (h: HEAP; size: CARDINAL) : ADDRESS ; */


/*
   Allocate - allocates, n, bytes of store from the heap.
*/

extern void *(MetaStore_Allocate(void *h, unsigned long n));


/*
   Deallocate - deallocates, n, bytes of store.
*/

extern void MetaStore_Deallocate(void *h, void *a, unsigned long size


/*
   ReAllocate - attempts to reallocate one chunk as two. It assumes t
                address, a, has already been allocated with fromsize
                Upon return there are two fragments allocated:

                a with tosize bytes
                b with bsize  bytes

                Note that tosize + bsize may not be equal to fromsize
                     as there is a system overhead.
*/
```

```
(*
   VerifyHeap - verifies that HEAP, h, is consistant. It checks that
                the sum of the freelist of usedlist equals the total
*)

PROCEDURE VerifyHeap (h: HEAP) ;


(*
   GetTotalSize - returns the total heap size.
*)

PROCEDURE GetTotalSize (h: HEAP) : CARDINAL ;


(*
   GetUsedSize - returns the amount of the heap that is used.
*)

PROCEDURE GetUsedSize (h: HEAP) : CARDINAL ;


(*
   GetFreeSize - returns the amount of the heap that is free.
*)

PROCEDURE GetFreeSize (h: HEAP) : CARDINAL ;


(*
   GetNoOfUsedFragments - returns the number of fragments held on the
*)
```



```
extern void MetaStore_Reallocate(void *h, void *a, unsigned long from
  unsigned long tosize, void **b, unsigned long *bsize);

/*
   GetLargestFree - returns the largest free space fragment size in b
                    in a heap. NOTE it does not allocate this fragmen
*/

extern unsigned long MetaStore_GetLargestFree(void *h);


/*
   VerifyHeap - verifies that HEAP, h, is consistant. It checks that
                the sum of the freelist of usedlist equals the total
*/

extern void MetaStore_VerifyHeap(void *h);


/*
   GetTotalSize - returns the total heap size.
*/

extern unsigned long MetaStore_GetTotalSize(void *h);


/*
   GetUsedSize - returns the amount of the heap that is used.
*/

extern unsigned long MetaStore_GetUsedSize(void *h);
```

```
PROCEDURE GetNoOfUsedFragments (h: HEAP) : CARDINAL ;

(*
   GetNoOfFreeFragments - returns the number of fragments held on the
*)

PROCEDURE GetNoOfFreeFragments (h: HEAP) : CARDINAL ;


END MetaStore.
```



```
/*
   GetFreeSize - returns the amount of the heap that is free.
*/

extern unsigned long MetaStore_GetFreeSize(void *h);


/*
   GetNoOfUsedFragments - returns the number of fragments held on the
*/

extern unsigned long MetaStore_GetNoOfUsedFragments(void *h);


/*
   GetNoOfFreeFragments - returns the number of fragments held on the
*/

extern unsigned long MetaStore_GetNoOfFreeFragments(void *h);


#undef vextern

#endif /*MetaStoreH*/

/* End. */
```

```
DEFINITION MODULE MonStrIO ;

(*
   Title      : MonStrIO
   Author     : Gaius Mulley
   Date       : 16/8/87
   LastEdit   : 12/8/94
   System     : LOGITECH MODULA-2/86 & (RTS GM2)
   Description: Provides a method of writting text to a screen that i
                unbuffered and process independant.
*)


FROM StdIO IMPORT ProcWrite ;

EXPORT QUALIFIED DebuggingStream,
                 WriteString,
                 WriteLn, WriteCard, WriteHex,
                 WriteBin, WriteInt,
                 Init ;



PROCEDURE DebuggingStream (p: ProcWrite) ;

PROCEDURE WriteLn ;

PROCEDURE WriteString (a: ARRAY OF CHAR) ;

PROCEDURE WriteCard (x, n: CARDINAL) ;
```



```
/* Header for module MonStrIO, generated by p2c */
#ifndef MonStrIOH
#define MonStrIOH


/*
   Title      : MonStrIO
   Author     : Gaius Mulley
   Date       : 16/8/87
   LastEdit   : 12/8/94
   System     : LOGITECH MODULA-2/86 & (RTS GM2)
   Description: Provides a method of writting text to a screen that i
                unbuffered and process independant.
*/


#ifndef StdIOH
#include "StdIO.h"
#endif


#ifdef MonStrIOG
# define vextern
#else
# define vextern extern
#endif
```

```
PROCEDURE WriteHex (x, n: CARDINAL) ;

PROCEDURE WriteBin (x, n: CARDINAL) ;

PROCEDURE WriteInt (x: INTEGER; n: CARDINAL) ;


(*
   Init - sets the DebuggingStream to use Scn.Write.
          Should only be called by M2RTS.
*)

PROCEDURE Init ;


END MonStrIO.
```



```
extern void MonStrIO_DebuggingStream(void (*p)(Char));

extern void MonStrIO_WriteLn(void);

extern void MonStrIO_WriteString(const long a_LOW, const long a_HIGH,
                                 const Char *a);

extern void MonStrIO_WriteCard(unsigned long x, unsigned long n);

extern void MonStrIO_WriteHex(unsigned long x, unsigned long n);

extern void MonStrIO_WriteBin(unsigned long x, unsigned long n);

extern void MonStrIO_WriteInt(long x, unsigned long n);


/*
   Init - sets the DebuggingStream to use Scn.Write.
          Should only be called by M2RTS.
*/

extern void MonStrIO_Init(void);


#undef vextern

#endif /*MonStrIOH*/

/* End. */
```

```
DEFINITION MODULE NonAscii ;


EXPORT QUALIFIED hm , up , puh, lft, rgt, end, dwn,
                 chm, cpu, bw , fw , ce , cpd, in ,
                 dl , pdh ;

CONST
   hm  = 107C ;  (* Home                        *)
   up  = 110C ;  (* Cursor Up                   *)
   puh = 111C ;  (* Page Up and Home            *)
   lft = 113C ;  (* Left Cursor                 *)
   rgt = 115C ;  (* Right Cursor                *)
   end = 117C ;  (* End                         *)
   dwn = 120C ;  (* Cursor Down                 *)
   in  = 122C ;  (* Insert                      *)
   dl  = 123C ;  (* Delete                      *)
   pdh = 121C ;  (* Page Down and Home          *)
   bw  = 163C ;  (* Backwards a Word            *)
   fw  = 164C ;  (* Forwards a Word             *)
   ce  = 165C ;  (* Ctrl End                    *)
   cpd = 166C ;  (* Ctrl Page Up                *)
   chm = 167C ;  (* Ctrl Home                   *)
```



```
/* Header for module NonAscii, generated by p2c */
#ifndef NonAsciiH
#define NonAsciiH


#ifdef NonAsciiG
# define vextern
#else
# define vextern extern
#endif



#define NonAscii_hm     'G'   /* Home                        */
#define NonAscii_up     'H'   /* Cursor Up                   */
#define NonAscii_puh    'I'   /* Page Up and Home            */
#define NonAscii_lft    'K'   /* Left Cursor                 */
#define NonAscii_rgt    'M'   /* Right Cursor                */
#define NonAscii_end    'O'   /* End                         */
#define NonAscii_dwn    'P'   /* Cursor Down                 */
#define NonAscii_in     'R'   /* Insert                      */
#define NonAscii_dl     'S'   /* Delete                      */
#define NonAscii_pdh    'Q'   /* Page Down and Home          */
#define NonAscii_bw     's'   /* Backwards a Word            */
#define NonAscii_fw     't'   /* Forwards a Word             */
```

```
   cpu = 204C ;  (* Ctrl Page Up                *)

END NonAscii.
```



```
#define NonAscii_ce     'u'   /* Ctrl End                   */
#define NonAscii_cpd    'v'   /* Ctrl Page Up               */
#define NonAscii_chm    'w'   /* Ctrl Home                  */
#define NonAscii_cpu    132   /* Ctrl Page Up               */

#undef vextern

#endif /*NonAsciiH*/

/* End. */
```

```
DEFINITION MODULE NumberIO ;

(*
   Author     : Gaius Mulley
   Date       : '84
   LastEdit   : Thu Nov 23 19:01:25 GMT 2000
   Description: Provides all the input/output of numbers, and also th
                of numbers to strings and visa versa.
*)

EXPORT QUALIFIED ReadCard, WriteCard, ReadHex, WriteHex, ReadInt, Wri
                 CardToStr, StrToCard, StrToHex, HexToStr, StrToInt,
                 ReadOct, WriteOct, OctToStr, StrToOct,
                 ReadBin, WriteBin, BinToStr, StrToBin,
                 StrToBinInt, StrToHexInt, StrToOctInt ;


PROCEDURE ReadCard (VAR x: CARDINAL) ;

PROCEDURE WriteCard (x, n: CARDINAL) ;

PROCEDURE ReadHex (VAR x: CARDINAL) ;

PROCEDURE WriteHex (x, n: CARDINAL) ;

PROCEDURE ReadInt (VAR x: INTEGER) ;

PROCEDURE WriteInt (x: INTEGER ; n: CARDINAL) ;

PROCEDURE CardToStr (x, n: CARDINAL ; VAR a: ARRAY OF CHAR) ;
```



```
/* Header for module NumberIO, generated by p2c */
#ifndef NumberIOH
#define NumberIOH


#ifdef NumberIOG
# define vextern
#else
# define vextern extern
#endif

/*
   Author     : Gaius Mulley
   Date       : '84
   LastEdit   : Thu Nov 23 19:01:25 GMT 2000
   Description: Provides all the input/output of numbers, and also th
                of numbers to strings and visa versa.
*/


extern void NumberIO_ReadCard(unsigned long *x);

extern void NumberIO_WriteCard(unsigned long x, unsigned long n);

extern void NumberIO_ReadHex(unsigned long *x);

extern void NumberIO_WriteHex(unsigned long x, unsigned long n);

extern void NumberIO_ReadInt(long *x);
```

```
PROCEDURE StrToCard (a: ARRAY OF CHAR ; VAR x: CARDINAL) ;

PROCEDURE HexToStr (x, n: CARDINAL ; VAR a: ARRAY OF CHAR) ;

PROCEDURE StrToHex (a: ARRAY OF CHAR ; VAR x: CARDINAL) ;

PROCEDURE IntToStr (x: INTEGER ; n: CARDINAL ; VAR a: ARRAY OF CHAR)

PROCEDURE StrToInt (a: ARRAY OF CHAR ; VAR x: INTEGER) ;

PROCEDURE ReadOct (VAR x: CARDINAL) ;

PROCEDURE WriteOct (x, n: CARDINAL) ;

PROCEDURE OctToStr (x, n: CARDINAL ; VAR a: ARRAY OF CHAR) ;

PROCEDURE StrToOct (a: ARRAY OF CHAR ; VAR x: CARDINAL) ;

PROCEDURE ReadBin (VAR x: CARDINAL) ;

PROCEDURE WriteBin (x, n: CARDINAL) ;

PROCEDURE BinToStr (x, n: CARDINAL ; VAR a: ARRAY OF CHAR) ;

PROCEDURE StrToBin (a: ARRAY OF CHAR ; VAR x: CARDINAL) ;

PROCEDURE StrToBinInt (a: ARRAY OF CHAR ; VAR x: INTEGER) ;

PROCEDURE StrToHexInt (a: ARRAY OF CHAR ; VAR x: INTEGER) ;

PROCEDURE StrToOctInt (a: ARRAY OF CHAR ; VAR x: INTEGER) ;


END NumberIO.
```



```
extern void NumberIO_WriteInt(long x, unsigned long n);

extern void NumberIO_CardToStr(unsigned long x, unsigned long n,
                               const long a_LOW, const long a_HIGH, Char *a);

extern void NumberIO_StrToCard(const long a_LOW, const long a_HIGH,
                               const Char *a, unsigned long *x);

extern void NumberIO_HexToStr(unsigned long x, unsigned long n,
                              const long a_LOW, const long a_HIGH, Char *a);

extern void NumberIO_StrToHex(const long a_LOW, const long a_HIGH,
                              const Char *a, unsigned long *x);

extern void NumberIO_IntToStr(long x, unsigned long n, const long a_L
                              const long a_HIGH, Char *a);

extern void NumberIO_StrToInt(const long a_LOW, const long a_HIGH,
                              const Char *a, long *x);

extern void NumberIO_ReadOct(unsigned long *x);

extern void NumberIO_WriteOct(unsigned long x, unsigned long n);

extern void NumberIO_OctToStr(unsigned long x, unsigned long n,
                              const long a_LOW, const long a_HIGH, Char *a);

extern void NumberIO_StrToOct(const long a_LOW, const long a_HIGH,
                              const Char *a, unsigned long *x);

extern void NumberIO_ReadBin(unsigned long *x);
```

```
DEFINITION MODULE OSParameters ;

(*
    Title      : OSParameters
    Author     : Gaius Mulley
    System     : UNIX (gm2)
    Date       : Mon Aug  1 11:44:46 1994
    Last edit  : Mon Aug  1 11:44:46 1994
    Description: defines the locations for the parameters to be
                 passed from the second boot stage to the OS.
                 These constants are used by second.mod and the
                 386/486 realtime system.
*)

EXPORT QUALIFIED ExtendedMemAddr,   VideoAddrAddr,
                 OSSizeAddr,        NoOfSectorsAddr,
                 DebuggingAddr,     StackSizeAddr ;


CONST
   ParametersToOS = 640 * 1024 ;

   ExtendedMemAddr  = ParametersToOS-2 ;
   VideoAddrAddr    = ParametersToOS-4 ;
   OSSizeAddr       = ParametersToOS-6 ;
```



```
/* Header for module OSParameters, generated by p2c */
#ifndef OSParametersH
#define OSParametersH


#ifdef OSParametersG
# define vextern
#else
# define vextern extern
#endif


/*
    Title      : OSParameters
    Author     : Gaius Mulley
    System     : UNIX (gm2)
    Date       : Mon Aug  1 11:44:46 1994
    Last edit  : Mon Aug  1 11:44:46 1994
    Description: defines the locations for the parameters to be
                 passed from the second boot stage to the OS.
                 These constants are used by second.mod and the
                 386/486 realtime system.
*/


#define OSParameters_ParametersToOS  655360
```

```
   NoOfSectorsAddr  = ParametersToOS-8 ;
   DebuggingAddr    = ParametersToOS-10 ;
   StackSizeAddr    = ParametersToOS-12 ;


END OSParameters.
```



```
#define OSParameters_ExtendedMemAddr  (OSParameters_ParametersToOS - 
#define OSParameters_VideoAddrAddr  (OSParameters_ParametersToOS - 4)
#define OSParameters_OSSizeAddr  (OSParameters_ParametersToOS - 6)
#define OSParameters_NoOfSectorsAddr  (OSParameters_ParametersToOS - 
#define OSParameters_DebuggingAddr  (OSParameters_ParametersToOS - 10
#define OSParameters_StackSizeAddr  (OSParameters_ParametersToOS - 12


#undef vextern

#endif /*OSParametersH*/

/* End. */
```

```
DEFINITION MODULE PortIO ;

(*
    Title      : PortIO
    Author     : Gaius Mulley
    System     : UNIX (gm2)
    Date       : Tue Jul 23 15:55:15 1996
    Last edit  : Tue Jul 23 15:55:15 1996
    Description: provides a separate module for In and Out instructio
*)

FROM SYSTEM IMPORT BYTE, WORD, ADDRESS ;

EXPORT QUALIFIED SlowDownIO, In8, Out8, In16, Out16, InS8, OutS8, InS


(*
   SlowDownIO - linux mechanism for slowing down port accesses.
*)

PROCEDURE SlowDownIO ;


(*
   In8 - returns a BYTE from port, Port.
*)

PROCEDURE In8 (Port: CARDINAL) : BYTE ;


(*
```



```
/* Header for module PortIO, generated by p2c */
#ifndef PortIOH
#define PortIOH


#ifndef SYSTEMH
#include "SYSTEM.h"
#endif


#ifdef PortIOG
# define vextern
#else
# define vextern extern
#endif

/*
    Title      : PortIO
    Author     : Gaius Mulley
    System     : UNIX (gm2)
    Date       : Tue Jul 23 15:55:15 1996
    Last edit  : Tue Jul 23 15:55:15 1996
    Description: provides a separate module for In and Out instructio
*/



/*
```

```
   Out8 - sends a byte, Value, to port, Port.
*)

PROCEDURE Out8 (Port: CARDINAL; Value: BYTE) ;


(*
   In16 - returns a WORD from port, Port.
          The top 16 bits are 0, the bottom 16 bits are assigned from
          the value of the port.
*)

PROCEDURE In16 (Port: CARDINAL) : WORD ;


(*
   Out16 - sends a 16 bit value, Value, to port, Port.
           Value is actually a 32 bit entity but the top
           16 bits are ignored.
*)

PROCEDURE Out16 (Port: CARDINAL; Value: WORD) ;


(*
   InS8 - reads, n, bytes in from port, Port, to address, a.
*)

PROCEDURE InS8 (Port: CARDINAL; a: ADDRESS; n: CARDINAL) ;


(*
```



```
   SlowDownIO - linux mechanism for slowing down port accesses.
*/

extern void PortIO_SlowDownIO(void);


/*
   In8 - returns a BYTE from port, Port.
*/

extern uchar PortIO_In8(unsigned long Port);


/*
   Out8 - sends a byte, Value, to port, Port.
*/

extern void PortIO_Out8(unsigned long Port, uchar Value);


/*
   In16 - returns a WORD from port, Port.
          The top 16 bits are 0, the bottom 16 bits are assigned from
          the value of the port.
*/

extern long PortIO_In16(unsigned long Port);


/*
   Out16 - sends a 16 bit value, Value, to port, Port.
           Value is actually a 32 bit entity but the top
```

```
   InS16 - reads, n, 16 bit words in from port, Port, to address, a.
*)

PROCEDURE InS16 (Port: CARDINAL; a: ADDRESS; n: CARDINAL) ;


(*
   InS32 - reads, n, 32 bit words in from port, Port, to address, a.
*)

PROCEDURE InS32 (Port: CARDINAL; a: ADDRESS; n: CARDINAL) ;


(*
   OutS8 - writes, n, bytes to port, Port, from address, a.
*)

PROCEDURE OutS8 (Port: CARDINAL; a: ADDRESS; n: CARDINAL) ;


(*
   OutS16 - writes, n, 16 bit words to port, Port, from address, a.
*)

PROCEDURE OutS16 (Port: CARDINAL; a: ADDRESS; n: CARDINAL) ;


(*
   OutS32 - writes, n, 32 bit words to port, Port, from address, a.
*)

PROCEDURE OutS32 (Port: CARDINAL; a: ADDRESS; n: CARDINAL) ;


END PortIO.
```



```
           16 bits are ignored.
*/

extern void PortIO_Out16(unsigned long Port, long Value);


/*
   InS8 - reads, n, bytes in from port, Port, to address, a.
*/

extern void PortIO_InS8(unsigned long Port, void *a, unsigned long n)


/*
   InS16 - reads, n, 16 bit words in from port, Port, to address, a.
*/

extern void PortIO_InS16(unsigned long Port, void *a, unsigned long n


/*
   InS32 - reads, n, 32 bit words in from port, Port, to address, a.
*/

extern void PortIO_InS32(unsigned long Port, void *a, unsigned long n


/*
   OutS8 - writes, n, bytes to port, Port, from address, a.
*/

extern void PortIO_OutS8(unsigned long Port, void *a, unsigned long n


/*
```

```
   OutS16 - writes, n, 16 bit words to port, Port, from address, a.
*/

extern void PortIO_OutS16(unsigned long Port, void *a, unsigned long

/*
   OutS32 - writes, n, 32 bit words to port, Port, from address, a.
*/

extern void PortIO_OutS32(unsigned long Port, void *a, unsigned long

#undef vextern

#endif /*PortIOH*/

/* End. */
```

```
DEFINITION MODULE SYSTEM ;

(*
   Author     : Gaius Mulley
   Title      : SYSTEM
   Date       : 3/4/86
   Description: Implements the SYSTEM dependant module
                in the Modula-2 compiler.
   Last update: 9/9/89  - types and pseudo procedures exported:
                          ADR, SIZE, TSIZE.
                29/7/94 - started to add items to SYSTEM.mod
                          added In, Out.
                3/8/94  - added TRANSFER
                4/8/94  - added NEWPROCESS
                10/8/94 - added IOTRANSFER
                12/8/94 - added LISTEN
                23/7/97 - removed port io and placed into separate mo

   $Version$

   $Log: SYSTEM.def,v $
   Revision 1.2  1996/07/26 12:05:09  gaius
   added minor support modules for the larger test MorlocTower
```



```
#ifndef SYSTEMH
#define SYSTEMH

/*
   Author     : Gaius Mulley
   Title      : SYSTEM
   Date       : 3/4/86
   Description: Implements the SYSTEM dependant module
                in the Modula-2 compiler.
   Last update: 9/9/89  - types and pseudo procedures exported:
                          ADR, SIZE, TSIZE.
                29/7/94 - started to add items to SYSTEM.mod
                          added In, Out.
                3/8/94  - added TRANSFER
                4/8/94  - added NEWPROCESS
                10/8/94 - added IOTRANSFER
                12/8/94 - added LISTEN
                23/7/97 - removed port io and placed into separate mo
*/


typedef enum {
  SYSTEM_On, SYSTEM_Off
```

```
*)


EXPORT QUALIFIED (* the following are built into the compiler: *)
                 ADDRESS, WORD, BYTE, BITSET, ADR, TSIZE, SIZE,

                 (* and the rest are implemented in SYSTEM.mod *)
                 OnOrOff, TurnInterrupts,
                 PROCESS, TRANSFER, NEWPROCESS, IOTRANSFER,
                 LISTEN, CheckOff,

                 (* Init is ONLY exported for M2RTS *)
                 Init ;


TYPE
   OnOrOff = (On, Off) ;
   PROCESS ;


(*
   TRANSFER - save the current volatile environment into, p1.
              Restore the volatile environment from, p2.
*)

PROCEDURE TRANSFER (VAR p1: PROCESS; p2: PROCESS) ;


(*
   NEWPROCESS - p is a parameterless procedure, a, is the origin of
                the workspace used for the process stack and containi
```



```
} SYSTEM_OnOrOff;


#ifndef SYSTEMG
typedef void *SYSTEM_PROCESS;  /* opaque type declaration */
#endif


/*
   TRANSFER - save the current volatile environment into, p1.
              Restore the volatile environment from, p2.
*/


extern void SYSTEM_TRANSFER(void **p1, void *p2);


/*
   NEWPROCESS - p is a parameterless procedure, a, is the origin of
                the workspace used for the process stack and containi
                the volatile environment of the process. n, is the am
                in bytes of this workspace. new, is the new process.
*/

extern void SYSTEM_NEWPROCESS(void (*p)(void), void *a, unsigned long
                  void **new_);


/*
   IOTRANSFER - saves the current volatile environment into, First,
                and restores volatile environment, Second.
```

```
               the volatile environment of the process. n, is the am
               in bytes of this workspace. new, is the new process.
*)

PROCEDURE NEWPROCESS (p: PROC; a: ADDRESS; n: CARDINAL; VAR new: PROC

(*
   IOTRANSFER - saves the current volatile environment into, First,
               and restores volatile environment, Second.
               When an interrupt, InterruptNo, is encountered then
               the reverse takes place. (The then current volatile
               environment is shelved onto Second and First is resum

               NOTE: that upon interrupt the Second might not be the
                     same process as that before the original call t
                     IOTRANSFER.
*)

PROCEDURE IOTRANSFER (VAR First, Second: PROCESS; InterruptNo: CARDIN

(*
   LISTEN - briefly listen for any interrupts.
*)

PROCEDURE LISTEN ;

(*
```



```
               When an interrupt, InterruptNo, is encountered then
               the reverse takes place. (The then current volatile
               environment is shelved onto Second and First is resum

               NOTE: that upon interrupt the Second might not be the
                     same process as that before the original call t
                     IOTRANSFER.
*/

extern void SYSTEM_IOTRANSFER(void **First, void **Second,
                              unsigned long InterruptNo);


/*
   LISTEN - briefly listen for any interrupts.
*/

extern void SYSTEM_LISTEN(void);


/*
   TurnInterrupts - switches interrupts on or off depending
                    on Switch. It returns the old value.
*/

extern SYSTEM_OnOrOff SYSTEM_TurnInterrupts(SYSTEM_OnOrOff Switch);


/*
   Init - initialize SYSTEM data structures. IOTRANSFER process table
```

```
   TurnInterrupts - switches interrupts on or off depending
                    on Switch. It returns the old value.
*)

PROCEDURE TurnInterrupts (Switch: OnOrOff) : OnOrOff ;


(*
   Init - initialize SYSTEM data structures. IOTRANSFER process table
*)

PROCEDURE Init ;


PROCEDURE CheckOff ;


END SYSTEM.
```



```
*/

extern void SYSTEM_Init(void);


extern void SYSTEM_CheckOff(void);

#endif /*SYSTEMH*/
```

```
DEFINITION MODULE Scn ;


EXPORT QUALIFIED Write, ClearScreen, MoveCursor,
                 DeleteLine, AppendLine, ClearLine,
                 Bell, Init ;


(*
   Write - writes character, ch, to the screen.
           Write interprets cr, lf.
*)

PROCEDURE Write (ch: CHAR) ;


(*
   MoveCursor - moves the cursor to position, x, y.
*)

PROCEDURE MoveCursor (x, y: CARDINAL) ;


(*
   ClearScreen - clears the screen, filling it with spaces.
*)

PROCEDURE ClearScreen ;


(*
   ClearLine - clears line, y, filling it with spaces.
*)
```



```
/* Header for module Scn, generated by p2c */
#ifndef ScnH
#define ScnH


#ifdef ScnG
# define vextern
#else
# define vextern extern
#endif



/*
   Write - writes character, ch, to the screen.
           Write interprets cr, lf.
*/

extern void Scn_Write(Char ch);


/*
   MoveCursor - moves the cursor to position, x, y.
*/

extern void Scn_MoveCursor(unsigned long x, unsigned long y);


/*
   ClearScreen - clears the screen, filling it with spaces.
*/
```

```
PROCEDURE ClearLine (y: CARDINAL) ;


(*
   AppendLine - adds another line after, y, to the display, all the
                other lines move down one line.
*)

PROCEDURE AppendLine (y: CARDINAL) ;


(*
   DeleteLine - deletes line, y, from the screen, the lower lines
                move up to fill in the gap.
*)

PROCEDURE DeleteLine (y: CARDINAL) ;


(*
   Bell - rings the bell on the console.
*)

PROCEDURE Bell ;


(*
   Init - initialize all module data structures.
*)

PROCEDURE Init ;


END Scn.
```



```
extern void Scn_ClearScreen(void);


/*
   ClearLine - clears line, y, filling it with spaces.
*/

extern void Scn_ClearLine(unsigned long y);


/*
   AppendLine - adds another line after, y, to the display, all the
                other lines move down one line.
*/

extern void Scn_AppendLine(unsigned long y);


/*
   DeleteLine - deletes line, y, from the screen, the lower lines
                move up to fill in the gap.
*/

extern void Scn_DeleteLine(unsigned long y);


/*
   Bell - rings the bell on the console.
*/

extern void Scn_Bell(void);


/*
```

```
   Init - initialize all module data structures.
*/

extern void Scn_Init(void);


#undef vextern

#endif /*ScnH*/

/* End. */
```

```
DEFINITION MODULE StdIO ;

(*
   Author     : Gaius Mulley
   Title      : StdIO
   Date       : 12/2/86  [$Date: 1997/11/03 17:21:43 $]
   SYSTEM     : UNIX SUN and Logitech M2
   Description: Exports a general Read and Write procedure that ALL c
                processes should use.
   Version    : $Revision: 1.4 $
*)

EXPORT QUALIFIED ProcRead, ProcWrite,
                 Read, Write, PushOutput, PopOutput, GetCurrentOutput



TYPE
   ProcWrite = PROCEDURE (CHAR) ;
   ProcRead  = PROCEDURE (VAR CHAR) ;


(*
   Read - is the generic procedure that all higher application layers
          should use to receive a character.
*)

PROCEDURE Read (VAR ch: CHAR) ;
```



```
/* Header for module StdIO, generated by p2c */
#ifndef StdIOH
#define StdIOH


#ifdef StdIOG
# define vextern
#else
# define vextern extern
#endif


/*
   Author     : Gaius Mulley
   Title      : StdIO
   Date       : 12/2/86  [$Date: 1997/11/03 17:21:43 $]
   SYSTEM     : UNIX SUN and Logitech M2
   Description: Exports a general Read and Write procedure that ALL c
                processes should use.
   Version    : $Revision: 1.4 $
*/


typedef void (*StdIO_ProcWrite)(Char);

typedef void (*StdIO_ProcRead)(Char *);


/*
```

```
(*
   Write - is the generic procedure that all higher application layer
           should use to emit a character.
*)

PROCEDURE Write (ch: CHAR) ;


(*
   PushOutput - pushes the current Write procedure onto a stack,
                any future references to Write will actually invoke
                procedure, p.
*)

PROCEDURE PushOutput (p: ProcWrite) ;


(*
   PopOutput - restores Write to use the previous output procedure.
*)

PROCEDURE PopOutput ;


(*
   GetCurrentOutput - returns the current output procedure.
*)

PROCEDURE GetCurrentOutput () : ProcWrite ;


(*
```



```
   Read - is the generic procedure that all higher application layers
          should use to receive a character.
*/


extern void StdIO_Read(Char *ch);


/*
   Write - is the generic procedure that all higher application layer
           should use to emit a character.
*/

extern void StdIO_Write(Char ch);


/*
   PushOutput - pushes the current Write procedure onto a stack,
                any future references to Write will actually invoke
                procedure, p.
*/

extern void StdIO_PushOutput(void (*p)(Char));


/*
   PopOutput - restores Write to use the previous output procedure.
*/

extern void StdIO_PopOutput(void);


/*
```

```
   Init - initialize this module data structures.
          This is only made available for M2RTS.
*)

PROCEDURE Init ;


END StdIO.
```



```
   GetCurrentOutput - returns the current output procedure.
*/

extern void (*(StdIO_GetCurrentOutput(void)))(void);


/*
   Init - initialize this module data structures.
          This is only made available for M2RTS.
*/

extern void StdIO_Init(void);


#undef vextern

#endif /*StdIOH*/

/* End. */
```

```
DEFINITION MODULE StrIO ;

(*
   Title      : StrIO
   Author     : Gaius Mulley
   Date       : 8/84
   LastEdit   : 27/2/88
   System     : LOGITECH MODULA-2/86
   Description: Provides simple string input output routines.
*)

EXPORT QUALIFIED ReadString, WriteString,
                 WriteLn ;


(*
   WriteLn - writes a carriage return and a newline
             character.
*)

PROCEDURE WriteLn ;


(*
   ReadString - reads a sequence of characters into a string.
                Line editing accepts Del, Ctrl H, Ctrl W and
                Ctrl U.
```



```
/* Header for module StrIO, generated by p2c */
#ifndef StrIOH
#define StrIOH


#ifdef StrIOG
# define vextern
#else
# define vextern extern
#endif

/*
   Title      : StrIO
   Author     : Gaius Mulley
   Date       : 8/84
   LastEdit   : 27/2/88
   System     : LOGITECH MODULA-2/86
   Description: Provides simple string input output routines.
*/



/*
   WriteLn - writes a carriage return and a newline
             character.
*/
```

```
*)

PROCEDURE ReadString (VAR a: ARRAY OF CHAR) ;


(*
   WriteString - writes a string to the default output.
*)

PROCEDURE WriteString (a: ARRAY OF CHAR) ;


END StrIO.
```



```
extern void StrIO_WriteLn(void);


/*
   ReadString - reads a sequence of characters into a string.
                Line editing accepts Del, Ctrl H, Ctrl W and
                Ctrl U.
*/

extern void StrIO_ReadString(const long a_LOW, const long a_HIGH, Cha


/*
   WriteString - writes a string to the default output.
*/

extern void StrIO_WriteString(const long a_LOW, const long a_HIGH,
                  const Char *a);



#undef vextern

#endif /*StrIOH*/

/* End. */
```

```
DEFINITION MODULE StrLib ;

(*
   Author     : Gaius Mulley
   Date       : '84
   LastEdit   : 28/1/97
   Description: Provides string manipulation
*)

EXPORT QUALIFIED StrLen, StrCopy, StrEqual, StrConCat, StrLess,
                 IsSubString, StrRemoveWhitePrefix ;


(*
   StrLess - returns TRUE if string, a, alphabetically occurs before
             string, b.
*)

PROCEDURE StrLess (a, b: ARRAY OF CHAR) : BOOLEAN ;


(*
   StrEqual - performs a = b on two strings.
*)

PROCEDURE StrEqual (a, b: ARRAY OF CHAR) : BOOLEAN ;


(*
   StrLen - returns the length of string, a.
```



```
/* Header for module StrLib, generated by p2c */
#ifndef StrLibH
#define StrLibH


#ifdef StrLibG
# define vextern
#else
# define vextern extern
#endif

/*
   Author     : Gaius Mulley
   Date       : '84
   LastEdit   : 28/1/97
   Description: Provides string manipulation
*/



/*
   StrLess - returns TRUE if string, a, alphabetically occurs before
             string, b.
*/

extern BOOLEAN StrLib_StrLess(const long a_LOW, const long a_HIGH,
                              const Char *a, const Char *b);
```

```
*)

PROCEDURE StrLen (a: ARRAY OF CHAR) : CARDINAL ;


(*
   StrCopy - effectively performs b := a with two strings.
*)

PROCEDURE StrCopy (a: ARRAY OF CHAR ; VAR b: ARRAY OF CHAR) ;


(*
   StrConCat - combines a and b into c.
*)

PROCEDURE StrConCat (a, b: ARRAY OF CHAR ; VAR c: ARRAY OF CHAR) ;


(*
   IsSubString - returns true if b is a subcomponent of a.
*)

PROCEDURE IsSubString (a, b: ARRAY OF CHAR) : BOOLEAN ;


(*
   StrRemoveWhitePrefix - copies string, into string, b, excluding an
                          space infront of a.
*)

PROCEDURE StrRemoveWhitePrefix (a: ARRAY OF CHAR; VAR b: ARRAY OF CHA


END StrLib.
```



```
/*
   StrEqual - performs a = b on two strings.
*/

extern BOOLEAN StrLib_StrEqual(const long a_LOW, const long a_HIGH,
                               const Char *a, const Char *b);


/*
   StrLen - returns the length of string, a.
*/

extern unsigned long StrLib_StrLen(const long a_LOW, const long a_HIG
                                   const Char *a);


/*
   StrCopy - effectively performs b := a with two strings.
*/

extern void StrLib_StrCopy(const long a_LOW, const long a_HIGH, const
                           const long b_LOW, const long b_HIGH, Char *b);


/*
   StrConCat - combines a and b into c.
*/

extern void StrLib_StrConCat(const long a_LOW, const long a_HIGH,
  const Char *a, const Char *b, const long c_LOW, const long c_HIGH,
```

```
/*
   IsSubString - returns true if b is a subcomponent of a.
*/

extern BOOLEAN StrLib_IsSubString(const long a_LOW, const long a_HIGH
                  const Char *a, const Char *b);


/*
   StrRemoveWhitePrefix - copies string, into string, b, excluding an
                          space infront of a.
*/

extern void StrLib_StrRemoveWhitePrefix(const long a_LOW, const long
  const Char *a, const long b_LOW, const long b_HIGH, Char *b);


#undef vextern

#endif /*StrLibH*/

/* End. */
```

```
DEFINITION MODULE SysStorage ;

(*
    Title      : SysStorage
    Author     : Gaius Mulley
    System     : UNIX (gm2)
    Date       : Fri May  2 15:19:49 1997
    Last edit  : Fri May  2 15:19:49 1997
    Description: provides dynamic allocation for the system component
                 of a realtime system. This allows the application to
                 use the traditional Storage module which can be
                 handled differently.
*)

FROM SYSTEM IMPORT ADDRESS ;

EXPORT QUALIFIED ALLOCATE, DEALLOCATE, Available, Init ;


(*
   ALLOCATE - attempt to allocate memory from the heap.
              NIL is returned in, a, if ALLOCATE fails.
*)

PROCEDURE ALLOCATE (VAR a: ADDRESS ; Size: CARDINAL) ;


(*
```



```
/* Header for module SysStorage, generated by p2c */
#ifndef SysStorageH
#define SysStorageH


#ifndef SYSTEMH
#include "SYSTEM.h"
#endif


#ifdef SysStorageG
# define vextern
#else
# define vextern extern
#endif

/*
    Title      : SysStorage
    Author     : Gaius Mulley
    System     : UNIX (gm2)
    Date       : Fri May  2 15:19:49 1997
    Last edit  : Fri May  2 15:19:49 1997
    Description: provides dynamic allocation for the system component
                 of a realtime system. This allows the application to
                 use the traditional Storage module which can be
```

```
   DEALLOCATE - return, Size, bytes to the heap.
*)

PROCEDURE DEALLOCATE (a: ADDRESS ; Size: CARDINAL) ;


(*
   Available - returns TRUE if, Size, bytes can be allocated.
*)

PROCEDURE Available (Size: CARDINAL) : BOOLEAN;


(*
   Init - initializes the heap.
*)

PROCEDURE Init ;


END SysStorage.
```



```
                 handled differently.
*/



/*
   ALLOCATE - attempt to allocate memory from the heap.
              NIL is returned in, a, if ALLOCATE fails.
*/

extern void SysStorage_ALLOCATE(void **a, unsigned long Size);


/*
   DEALLOCATE - return, Size, bytes to the heap.
*/

extern void SysStorage_DEALLOCATE(void *a, unsigned long Size);


/*
   Available - returns TRUE if, Size, bytes can be allocated.
*/

extern BOOLEAN SysStorage_Available(unsigned long Size);


/*
   Init - initializes the heap.
*/

extern void SysStorage_Init(void);


#undef vextern
```

```
DEFINITION MODULE TTIO ;

(*
    Title      : TTIO
    Author     : Gaius Mulley
    System     : UNIX (gm2)
    Date       : Thu Aug 25 13:25:50 1994
    Last edit  : Thu Aug 25 13:25:50 1994
    Description: provides a buffered Read from an interrupt
                 driven keyboard driver.
*)

EXPORT QUALIFIED Read,
                 SwitchLeds,
                 SwitchScroll, SwitchNum, SwitchCaps ;


PROCEDURE Read (VAR ch: CHAR) ;


(*
   SwitchLeds - switch the keyboard LEDs to the state defined
                by the BOOLEAN variables. TRUE = ON.
*)

PROCEDURE SwitchLeds (NumLock, CapsLock, ScrollLock: BOOLEAN) ;


(*
```



```
/* Header for module TTIO, generated by p2c */
#ifndef TTIOH
#define TTIOH


#ifdef TTIOG
# define vextern
#else
# define vextern extern
#endif

/*
    Title      : TTIO
    Author     : Gaius Mulley
    System     : UNIX (gm2)
    Date       : Thu Aug 25 13:25:50 1994
    Last edit  : Thu Aug 25 13:25:50 1994
    Description: provides a buffered Read from an interrupt
                 driven keyboard driver.
*/



extern void TTIO_Read(Char *ch);


/*
   SwitchLeds - switch the keyboard LEDs to the state defined
```

```
   SwitchScroll - switchs the scroll LED on or off.
*)

PROCEDURE SwitchScroll (Scroll: BOOLEAN) ;


(*
   SwitchNum - switches the Num LED on or off.
*)

PROCEDURE SwitchNum (Num: BOOLEAN) ;


(*
   SwitchCaps - switches the Caps LED on or off.
*)

PROCEDURE SwitchCaps (Caps: BOOLEAN) ;


END TTIO.
```



```
                by the BOOLEAN variables. TRUE = ON.
*/

extern void TTIO_SwitchLeds(BOOLEAN NumLock, BOOLEAN CapsLock,
                            BOOLEAN ScrollLock);


/*
   SwitchScroll - switchs the scroll LED on or off.
*/

extern void TTIO_SwitchScroll(BOOLEAN Scroll);


/*
   SwitchNum - switches the Num LED on or off.
*/

extern void TTIO_SwitchNum(BOOLEAN Num);


/*
   SwitchCaps - switches the Caps LED on or off.
*/

extern void TTIO_SwitchCaps(BOOLEAN Caps);


#undef vextern

#endif /*TTIOH*/

/* End. */
```

```
DEFINITION MODULE TimerHandler ;

(*
    Title      : TimerHandler
    Author     : Gaius Mulley
    System     : UNIX (gm2)
    Date       : Mon Oct 23 17:24:49 1995
    Last edit  : Mon Oct 23 17:24:49 1995
    Description: provides a simple timer handler for the
                 Realtime Executive.
                 It also provides the Executive with a basic
                 round robin scheduler.
*)

EXPORT QUALIFIED TicksPerSecond, GetTicks,
                 EVENT,
                 Sleep, ArmEvent, WaitOn, Cancel, ReArmEvent ;


CONST
   TicksPerSecond =   100 ;  (* Number of ticks per second.        *

TYPE
   EVENT ;


(*
```



```
/* Header for module TimerHandler, generated by p2c */
#ifndef TimerHandlerH
#define TimerHandlerH


#ifdef TimerHandlerG
# define vextern
#else
# define vextern extern
#endif


/*
    Title      : TimerHandler
    Author     : Gaius Mulley
    System     : UNIX (gm2)
    Date       : Mon Oct 23 17:24:49 1995
    Last edit  : Mon Oct 23 17:24:49 1995
    Description: provides a simple timer handler for the
                 Realtime Executive.
                 It also provides the Executive with a basic
                 round robin scheduler.
*/


#define TimerHandler_TicksPerSecond  100
```

```
   GetTicks - returns the number of ticks since boottime.
*)

PROCEDURE GetTicks () : CARDINAL ;


(*
   Sleep - suspends the current process for a time, t.
           The time is measured in ticks.
*)

PROCEDURE Sleep (t: CARDINAL) ;


(*
   ArmEvent - initializes an event, e, to occur at time, t.
              The time, t, is measured in ticks.
              The event is NOT placed onto the event queue.
*)

PROCEDURE ArmEvent (t: CARDINAL) : EVENT ;


(*
   WaitOn - places event, e, onto the event queue and then the callin
            process suspends. It is resumed up by either the event
            expiring or the event, e, being cancelled.
            TRUE is returned if the event was cancelled
            FALSE is returned if the event expires.
```



```
    /* Number of ticks per second.          */


#ifndef TimerHandlerG
/* opaque type declaration */
typedef void *TimerHandler_EVENT;

#endif


/*
   GetTicks - returns the number of ticks since boottime.
*/


extern unsigned long TimerHandler_GetTicks(void);


/*
   Sleep - suspends the current process for a time, t.
           The time is measured in ticks.
*/

extern void TimerHandler_Sleep(unsigned long t);


/*
   ArmEvent - initializes an event, e, to occur at time, t.
              The time, t, is measured in ticks.
              The event is NOT placed onto the event queue.
*/

extern void *(TimerHandler_ArmEvent(unsigned long t));
```

```
*)

PROCEDURE WaitOn (e: EVENT) : BOOLEAN ;


(*
   Cancel - cancels the event, e, on the event queue and makes
            the appropriate process runnable again.
            TRUE is returned if the event was cancelled and
            FALSE is returned is the event was not found or
                  no process was waiting on this event.
*)

PROCEDURE Cancel (e: EVENT) : BOOLEAN ;


(*
   ReArmEvent - removes an event, e, from the event queue. A new time
                is given to this event and it is then re-inserted ont
                event queue in the correct place.
                TRUE is returned if this occurred
                FALSE is returned if the event was not found.
*)

PROCEDURE ReArmEvent (e: EVENT; t: CARDINAL) : BOOLEAN ;


END TimerHandler.
```



```
/*
   WaitOn - places event, e, onto the event queue and then the callin
            process suspends. It is resumed up by either the event
            expiring or the event, e, being cancelled.
            TRUE is returned if the event was cancelled
            FALSE is returned if the event expires.
*/

extern BOOLEAN TimerHandler_WaitOn(void *e);


/*
   Cancel - cancels the event, e, on the event queue and makes
            the appropriate process runnable again.
            TRUE is returned if the event was cancelled and
            FALSE is returned is the event was not found or
                  no process was waiting on this event.
*/

extern BOOLEAN TimerHandler_Cancel(void *e);


/*
   ReArmEvent - removes an event, e, from the event queue. A new time
                is given to this event and it is then re-inserted ont
                event queue in the correct place.
```

```
DEFINITION MODULE WindowDevice ;

(*
   Author      : Gaius Mulley
   Version     : 1.1
   Date        : 4/3/86
   Description : Window Utilities
   Last Update : 12/7/86
                 4/8/94
*)


EXPORT QUALIFIED Window,
                 InitWindow, KillWindow, SetWindow,
                 SubWindow, AddWindow, NulWindow,
                 NoCursor, SmallCursor, LargeCursor,
                 MoveCursor, ClearWindow, MoveWindow,
                 WriteChar, WriteString, WriteLn,
                 DefaultWindow, SetDefaultWindow,
                 ColourWindow, SizeWindow, TitleWindow,
                 PutOnTop, PutOnBottom, SelectWindow ;


TYPE
   Window ;
```



```
/* Header for module WindowDevice, generated by p2c */
#ifndef WindowDeviceH
#define WindowDeviceH


#ifdef WindowDeviceG
# define vextern
#else
# define vextern extern
#endif


/*
   Author      : Gaius Mulley
   Version     : 1.1
   Date        : 4/3/86
   Description : Window Utilities
   Last Update : 12/7/86
                 4/8/94
*/



#ifndef WindowDeviceG
/* opaque type declaration */
typedef void *WindowDevice_Window;

#endif
```

```
(*
   InitWindow - returns a Window handle. This Window is uninitialized
*)

PROCEDURE InitWindow () : Window ;


(*
   KillWindow - destroys a Window, w, and returns NIL.
*)

PROCEDURE KillWindow (w: Window) : Window ;


(*
   SetWindow - sets a Window, w, to contain background colour, bg,
               foreground colour, fg. The width, height are specified
               and border indicates whether the window has a Grey bor
               The Window, w, is returned.
*)

PROCEDURE SetWindow (w: Window; bg, fg: CARDINAL;
                     width, height, x, y: CARDINAL;
                     border: BOOLEAN) : Window ;

(*
   AddWindow - adds a Window, w, to the display.
*)
```



```
/*
   InitWindow - returns a Window handle. This Window is uninitialized
*/


extern void *(WindowDevice_InitWindow(void));


/*
   KillWindow - destroys a Window, w, and returns NIL.
*/

extern void *(WindowDevice_KillWindow(void *w));


/*
   SetWindow - sets a Window, w, to contain background colour, bg,
               foreground colour, fg. The width, height are specified
               and border indicates whether the window has a Grey bor
               The Window, w, is returned.
*/

extern void *(WindowDevice_SetWindow(void *w, unsigned long bg,
        unsigned long fg, unsigned long width, unsigned long height,
        unsigned long x, unsigned long y, BOOLEAN border));

/*
   AddWindow - adds a Window, w, to the display.
*/
```

```
PROCEDURE AddWindow (w: Window) ;


(*
   SubWindow - subtracts a Window, w, from the display.
*)

PROCEDURE SubWindow (w: Window) ;


(*
   NulWindow - returns TRUE if Window, w, is NIL.
               (Meaning it does not exist).
*)

PROCEDURE NulWindow (w: Window) : BOOLEAN ;


(*
   MoveCursor - moves the cursor of Window, w, to x, y.
*)

PROCEDURE MoveCursor (w: Window; x, y: CARDINAL) ;

PROCEDURE LargeCursor (w: Window) ;

PROCEDURE SmallCursor (w: Window) ;

PROCEDURE NoCursor (w: Window) ;

PROCEDURE ClearWindow (w: Window) ;


(*
   MoveWindow - moves a Window, w, to position, x, y.
```



```
extern void WindowDevice_AddWindow(void *w);


/*
   SubWindow - subtracts a Window, w, from the display.
*/

extern void WindowDevice_SubWindow(void *w);


/*
   NulWindow - returns TRUE if Window, w, is NIL.
               (Meaning it does not exist).
*/

extern BOOLEAN WindowDevice_NulWindow(void *w);


/*
   MoveCursor - moves the cursor of Window, w, to x, y.
*/

extern void WindowDevice_MoveCursor(void *w, unsigned long x, unsigne

extern void WindowDevice_LargeCursor(void *w);

extern void WindowDevice_SmallCursor(void *w);

extern void WindowDevice_NoCursor(void *w);

extern void WindowDevice_ClearWindow(void *w);


/*
   MoveWindow - moves a Window, w, to position, x, y.
```

```
                The Window must have been removed from the display
                by SubWindow before this is called.
*)

PROCEDURE MoveWindow (w: Window; x, y: CARDINAL) ;


(*
   WriteChar - writes a character, ch, to Window, w.
*)

PROCEDURE WriteChar (w: Window; ch: CHAR) ;


(*
   WriteString - writes a string, a, to window, w.
*)

PROCEDURE WriteString (w: Window; a: ARRAY OF CHAR) ;


(*
   WriteLn - places the cursor onto the beginning of a new line
             in Window, w.
*)

PROCEDURE WriteLn (w: Window) ;

(*
   ColourWindow - alters the foreground and background
                  colour of Window, w.
```



```
                The Window must have been removed from the display
                by SubWindow before this is called.
*/

extern void WindowDevice_MoveWindow(void *w, unsigned long x, unsigne


/*
   WriteChar - writes a character, ch, to Window, w.
*/

extern void WindowDevice_WriteChar(void *w, Char ch);


/*
   WriteString - writes a string, a, to window, w.
*/

extern void WindowDevice_WriteString(void *w, const long a_LOW,
                                     const long a_HIGH, const Char *a);


/*
   WriteLn - places the cursor onto the beginning of a new line
             in Window, w.
*/

extern void WindowDevice_WriteLn(void *w);

/*
   ColourWindow - alters the foreground and background
```

```
*)

PROCEDURE ColourWindow (w: Window; bg, fg: CARDINAL) ;


(*
   SizeWindow - not implemented.
*)

PROCEDURE SizeWindow (w: Window; width, height: CARDINAL) ;


(*
   PutOnTop - places window, w, on top of the pile of windows.
*)

PROCEDURE PutOnTop (w: Window) ;


(*
   PutOnBottom - places Window, w, on the bottom of the pile
                 of Windows.
*)

PROCEDURE PutOnBottom (w: Window) ;

(*
   SelectWindow - returns a Window which can be seen at screen
                  location, x, y.
                  If no Window is seen then NIL is returned.
*)
```



```
                  colour of Window, w.
*/

extern void WindowDevice_ColourWindow(void *w, unsigned long bg,
                                      unsigned long fg);


/*
   SizeWindow - not implemented.
*/

extern void WindowDevice_SizeWindow(void *w, unsigned long width,
                                    unsigned long height);


/*
   PutOnTop - places window, w, on top of the pile of windows.
*/

extern void WindowDevice_PutOnTop(void *w);


/*
   PutOnBottom - places Window, w, on the bottom of the pile
                 of Windows.
*/

extern void WindowDevice_PutOnBottom(void *w);

/*
   SelectWindow - returns a Window which can be seen at screen
```

```
PROCEDURE SelectWindow (x, y: CARDINAL) : Window ;


(*
   TitleWindow - adds a title to a Window, w.
*)

PROCEDURE TitleWindow (w: Window; a: ARRAY OF CHAR) ;


(*
   DefaultWindow - returns the default window.
*)

PROCEDURE DefaultWindow () : Window ;


(*
   SetDefaultWindow - sets the default window.
*)

PROCEDURE SetDefaultWindow (w: Window) ;


END WindowDevice.
```



```
                    location, x, y.
                    If no Window is seen then NIL is returned.
*/

extern void *(WindowDevice_SelectWindow(unsigned long x, unsigned lon


/*
   TitleWindow - adds a title to a Window, w.
*/

extern void WindowDevice_TitleWindow(void *w, const long a_LOW,
                        const long a_HIGH, const Char *a);


/*
   DefaultWindow - returns the default window.
*/

extern void *(WindowDevice_DefaultWindow(void));


/*
   SetDefaultWindow - sets the default window.
*/

extern void WindowDevice_SetDefaultWindow(void *w);


#undef vextern

#endif /*WindowDeviceH*/

/* End. */
```

```
DEFINITION MODULE WordSizes ;

(*
    Title      : WordSizes
    Author     : Gaius Mulley
    System     : UNIX (gm2)
    Date       : Wed Jun 15 12:01:35 1994
    Last edit  : Wed Jun 15 12:01:35 1994
    Description: provides procedures for manipulating word sizes.
*)

FROM SYSTEM IMPORT BYTE ;

EXPORT QUALIFIED SHORTWORD, LONGWORD,
                 ShortWordToCardinal, LongWordToCardinal,
                 CardinalToShortWord, CardinalToLongWord,
                 FlipShortWord, FlipLongWord,
                 WriteLong, WriteShort ;


TYPE
   SHORTWORD = RECORD
                  ByteLo, ByteHi: BYTE ;
               END ;

   LONGWORD = RECORD
```



```
/* Header for module WordSizes, generated by p2c */
#ifndef WordSizesH
#define WordSizesH


#ifndef SYSTEMH
#include "SYSTEM.h"
#endif


#ifdef WordSizesG
# define vextern
#else
# define vextern extern
#endif


/*
    Title      : WordSizes
    Author     : Gaius Mulley
    System     : UNIX (gm2)
    Date       : Wed Jun 15 12:01:35 1994
    Last edit  : Wed Jun 15 12:01:35 1994
    Description: provides procedures for manipulating word sizes.
*/



typedef struct WordSizes_SHORTWORD {
```

```
    WordLo: SHORTWORD ;
    WordHi: SHORTWORD ;
  END ;


(*
   ShortWordToCardinal - converts a short word into a cardinal.
*)

PROCEDURE ShortWordToCardinal (s: SHORTWORD) : CARDINAL ;


(*
   LongWordToCardinal - converts a short word into a cardinal.
*)

PROCEDURE LongWordToCardinal (l: LONGWORD) : CARDINAL ;


(*
   CardinalToShortWord - converts a cardinal into a short word.
*)

PROCEDURE CardinalToShortWord (c: CARDINAL) : SHORTWORD ;


(*
   CardinalToLongWord - converts a cardinal into a long word.
*)

PROCEDURE CardinalToLongWord (c: CARDINAL) : LONGWORD ;


(*
```



```
  uchar ByteLo, ByteHi;
} WordSizes_SHORTWORD;

typedef struct WordSizes_LONGWORD {
  WordSizes_SHORTWORD WordLo, WordHi;
} WordSizes_LONGWORD;


/*
   ShortWordToCardinal - converts a short word into a cardinal.
*/


extern unsigned long WordSizes_ShortWordToCardinal(WordSizes_SHORTWOR


/*
   LongWordToCardinal - converts a short word into a cardinal.
*/

extern unsigned long WordSizes_LongWordToCardinal(WordSizes_LONGWORD


/*
   CardinalToShortWord - converts a cardinal into a short word.
*/

extern WordSizes_SHORTWORD WordSizes_CardinalToShortWord(unsigned lon


/*
   CardinalToLongWord - converts a cardinal into a long word.
*/
```

```
   FlipShortWord - returns the byte flipped SHORTWORD, s.
*)

PROCEDURE FlipShortWord (s: SHORTWORD) : SHORTWORD ;


(*
   FlipLongWord - returns the byte flipped LONGWORD, l.
*)

PROCEDURE FlipLongWord (l: LONGWORD) : LONGWORD ;


(*
   WriteLong - writes the long out in hex characters.
*)

PROCEDURE WriteLong (l: LONGWORD) ;


(*
   WriteShort - writes the short out in hex characters.
*)

PROCEDURE WriteShort (s: SHORTWORD) ;


END WordSizes.
```



```
extern WordSizes_LONGWORD WordSizes_CardinalToLongWord(unsigned long

/*
   FlipShortWord - returns the byte flipped SHORTWORD, s.
*/

extern WordSizes_SHORTWORD WordSizes_FlipShortWord(WordSizes_SHORTWOR


/*
   FlipLongWord - returns the byte flipped LONGWORD, l.
*/

extern WordSizes_LONGWORD WordSizes_FlipLongWord(WordSizes_LONGWORD l


/*
   WriteLong - writes the long out in hex characters.
*/

extern void WordSizes_WriteLong(WordSizes_LONGWORD l);


/*
   WriteShort - writes the short out in hex characters.
*/

extern void WordSizes_WriteShort(WordSizes_SHORTWORD s);

#undef vextern

#endif /*WordSizesH*/

/* End. */
```

```
DEFINITION MODULE crt0 ;

(*
    Title      : crt0
    Author     : Gaius Mulley
    System     : UNIX (gm2)
    Date       : Wed Aug  9 12:17:01 1995
    Last edit  : Wed Aug  9 12:17:01 1995
    Description: provides the startup routines for the realtime syste
*)

FROM SYSTEM IMPORT ADDRESS ;

EXPORT QUALIFIED RunProgram, red ;


(*
   RunProgram - runs the program held at address, CodeStart, length,
                It moves the program to, PositionOfStart, and runs it
                there. It also blows away any previous program
                (ie this program).
*)

PROCEDURE RunProgram (CodeStart       : ADDRESS;
                      LengthOfTextData: CARDINAL;
                      LengthOfBss     : CARDINAL;
```



```
/* Header for module crt0, generated by p2c */
#ifndef crt0H
#define crt0H


#ifndef SYSTEMH
#include "SYSTEM.h"
#endif


#ifdef crt0G
# define vextern
#else
# define vextern extern
#endif

/*
    Title      : crt0
    Author     : Gaius Mulley
    System     : UNIX (gm2)
    Date       : Wed Aug  9 12:17:01 1995
    Last edit  : Wed Aug  9 12:17:01 1995
    Description: provides the startup routines for the realtime syste
*/



/*
```

```
                    PositionOfStart : ADDRESS) ;

(*
   red - a debugging routine to display a red box at the top left
         of a colour screen (B8000H).
*)

PROCEDURE red ;


END crt0.
```



```
   RunProgram - runs the program held at address, CodeStart, length,
                It moves the program to, PositionOfStart, and runs it
                there. It also blows away any previous program
                (ie this program).
*/

extern void crt0_RunProgram(void *CodeStart, unsigned long LengthOfTe
                unsigned long LengthOfBss, void *PositionOfStart);

/*
   red - a debugging routine to display a red box at the top left
         of a colour screen (B8000H).
*/

extern void crt0_red(void);


#undef vextern

#endif /*crt0H*/

/* End. */
```

```
DEFINITION MODULE gdb ;

(*
    Title      : gdb
    Author     : Gaius Mulley
    System     : UNIX (gm2)
    Date       : Sat Aug  5 13:21:35 1995
    Last edit  : Sat Aug  5 13:21:35 1995
    Description: provides a Modula-2 interface to the i386-stub.c (re
                 If you want to debug the realtime system then a modu
                 IMPORT gdb ; gdb.Init ;
*)

EXPORT QUALIFIED Init, breakpoint ;


(*
   Init - initializes the remote debugging mechanism.
*)

PROCEDURE Init ;


(*
   breakpoint - calls gdb from within your code (simulates a breakpoi
*)

PROCEDURE breakpoint ;
```



```
/* Header for module gdb, generated by p2c */
#ifndef gdbH
#define gdbH


#ifdef gdbG
# define vextern
#else
# define vextern extern
#endif

/*
    Title      : gdb
    Author     : Gaius Mulley
    System     : UNIX (gm2)
    Date       : Sat Aug  5 13:21:35 1995
    Last edit  : Sat Aug  5 13:21:35 1995
    Description: provides a Modula-2 interface to the i386-stub.c (re
                 If you want to debug the realtime system then a modu
                 IMPORT gdb ; gdb.Init ;
*/



/*
   Init - initializes the remote debugging mechanism.
```

```
END gdb.
```



```
*/

extern void gdb_Init(void);


/*
   breakpoint - calls gdb from within your code (simulates a breakpoi
*/

extern void gdb_breakpoint(void);


#undef vextern

#endif /*gdbH*/

/* End. */
```

```
DEFINITION MODULE libc ;

(*
    Title      : libc
    Author     : Gaius Mulley
    System     : UNIX (gm2)
    Date       : Tue Jul 29 09:47:23 1997
    Last edit  : Tue Jul 29 09:47:23 1997
    Description: provides a simple restrictive libc interface for
                 the basic realtime system.
*)

EXPORT QUALIFIED exit ;


(*
   exit - does nothing and enters an infinite loop.
          This function is here to provide compatibility
          for other modules.
*)

PROCEDURE exit (r: INTEGER) ;


END libc.
```



```
/* Header for module libc, generated by p2c */
#ifndef libcH
#define libcH


#ifdef libcG
# define vextern
#else
# define vextern extern
#endif

/*
    Title      : libc
    Author     : Gaius Mulley
    System     : UNIX (gm2)
    Date       : Tue Jul 29 09:47:23 1997
    Last edit  : Tue Jul 29 09:47:23 1997
    Description: provides a simple restrictive libc interface for
                 the basic realtime system.
*/



/*
   exit - does nothing and enters an infinite loop.
          This function is here to provide compatibility
```

```
DEFINITION MODULE libg ;

(*
    Title      : libg
    Author     : Gaius Mulley
    System     : UNIX (gm2)
    Date       : Sat Aug  5 20:14:54 1995
    Last edit  : Sat Aug  5 20:14:54 1995
    Description: provides very simple mechanism to communicate with G
*)

EXPORT QUALIFIED Init, Read, Write ;


(*
   Init - initializes the libg module.
*)

PROCEDURE Init ;


PROCEDURE Read (VAR ch: CHAR) ;
PROCEDURE Write (ch: CHAR) ;


END libg.
```



```
/* Header for module libg, generated by p2c */
#ifndef libgH
#define libgH


#ifdef libgG
# define vextern
#else
# define vextern extern
#endif

/*
    Title      : libg
    Author     : Gaius Mulley
    System     : UNIX (gm2)
    Date       : Sat Aug  5 20:14:54 1995
    Last edit  : Sat Aug  5 20:14:54 1995
    Description: provides very simple mechanism to communicate with G
*/



/*
   Init - initializes the libg module.
*/

extern void libg_Init(void);
```

```
(*
   MainStackSize - returns the size of mains stack.
*)

PROCEDURE MainStackSize () : CARDINAL ;


(*
   Init - retrieve all parameters from second bootstage.
*)

PROCEDURE Init ;


(*
   ResetParameters - replaces the parameters back to the original pos
                     ready for a reboot.
*)

PROCEDURE ResetParameters ;


END osinit..di
```



```
DEFINITION MODULE osinit ;

(*
    Title      : osinit
    Author     : Gaius Mulley
    System     : UNIX (gm2)
    Date       : Wed Aug  3 13:27:10 1994
    Last edit  : Wed Aug  3 13:27:10 1994
    Description: provides an interface which contains all
                 the initialization constants for the realtime system
*)

FROM SYSTEM IMPORT ADDRESS ;

EXPORT QUALIFIED ScreenAddress,
```