THE ATARI FORUM

A Dialog For Atari Software Developers

Volume I, Number 3 **July/August 1988**

DEVELOPING MARKETS

by Joe Ferrari
Director, Applications Software

During the recently held show of the National Association of Music Merchants (NAMM), a MIDI (Musical Instruments Digital Interface) Developer's Council was formed. The principal reason for the formation of this group is to ensure that Atari maintains or expands its market share in the music market. As some of you may know, within the past year, the Atari ST has made significant penetration into this large niche. If this council is successful, I would like to extend this concept into other niches, such as desktop publishing, CAD, etc. If anyone has suggestions, please let me know.

A market where Atari can realize significant market share is in desktop publishing. At the present time we are working on several projects (long and short term) that will provide us with the neccessary tools to be competitive in this segment of the market. Issues such as fonts, PostScript compatibility, full-page displays etc., are all being addressed and in the next issue of this letter, we will cover some of those projects with an appeal for your support.

As all of you may be aware, during the last year, Atari has not had much of a presence in the US market; it is not due to Atari's lack of interest, on the contrary; due to forces beyond our control we had to abandon our plans. (The DRAM shortage caused limited product availability; this forced Atari to place emphasis in Europe—where we have been highly successful.) We believe that this problem is beginning to subside and we are now planning Atari's renaissance in the US market. At COMDEX in Las Vegas, we have taken additional space to stage this comeback. I hope you will be able to join us in this exciting event.

by Elizabeth Shook
Newsletters Editor

From big-name support by companies such as WordPerfect, to Atari's faithful developers of Atari-only product lines, Atari computers are backed by a variety of inexpensive, high-quality applications packages and solutions.

As Atari identifies and approaches strategic markets within the personal computer industry, we would like to encourage developers to consider looking more closely at products for which there appear to be a real need.

Established Markets

The strongest of these target markets is music, education, and desktop video. Atari proved its strength in the music market at the NAMM show in Atlanta last month. "The quality and diversity of MIDI software at the booth eloquently described our strength in the market," said Joe Ferrari, director of applications software. Atari equipment was prevalent thoughout the exhibition hall, and more and more celebrity musicians—most recently Hall & Oates—have become Atari converts.

In the educational market, Atari receives attention at the university level. And as a result of the efforts by dedicated users and dealers, Ataris are beginning to find their way into elementary and secondary schools. VARs such as Computer Curriculum Corporation (CCC) have comfortably settled themselves in their chosen market niche.

Antic Publishing Company is distributing a full line of desktop video software, successfully establishing Atari in the desktop video market. Computer graphics and animation combine television and video technology to produce spectacular effects.

Atari itself has made a splash in the desktop publishing market segment. The MEGA and SLM804 Laser system was featured on the cover of the national monthly, *Computer Shopper,* and appears in a review in *Personal Publishing* magazine. A current dealer promotion makes MEGA/Laser bundles available to users at special prices.

New Hardware

With development of high-powered workstations such as the new 32-bit line and the Transputer, there is more need than ever before for sophisticated applications software. The processing power of these systems

Continued on page 2

The Atari Forum was created—in its entirety—using an Atari Publishing System: MEGA 4, SLM804, Atari Deskset, Microsoft Write, and Easy-Draw.

Q&A

by John Feagans
Director of Software Technology

Here are the latest questions from the Atari developers' mailbag. Leave questions on Compuserve for PIN 70007,1072 or GO PCS57 for Atari developer SIG information.

BIOS

Q: I have a program which will work on the UK and US versions of the ST but does not work on the French. I am looking for a control-Q but never receive a match on the code.

A: It sounds like you are trying to match the complete long word containing the scan code for each key combination in which you are interested. Scan codes for the same ASCII code can change between country versions of the ST. The French use an AZERTY keyboard instead of QWERTY. Thus, the ASCII code will remain the same but the scan code, or key position, will be changed. The best procedure to follow here is only to use the scan code value if the ASCII value is zero. The ASCII code is zero for all cursor pad keys. Match on the ASCII codes whenever possible and your program will be assured of working from country to country.

VDI

Q: When I use the exchange mouse vector to insert my own handler for mouse position, where are the mouse x and mouse y positions passed to me?

A: This is information that is processor specific to GEM and is contained in an appendix to the documentation. Specifically, the x position is passed in D0 and the y position is passed in D1.

AES

Q: Sometimes my mouse exhibits some peculiar behavior in the application that I am writing. None of the parts of my windows are active the first time until I click on the desktop. The control panel will not close until I double click. Other times my menus will not pop down until I click on something else. What is wrong with the mouse?

A: After personally looking at your code I determined that the problem was with the graf_mouse call in which you turn the mouse on and off—you were doing it before the open virtual workstation call in your program. Later, after the open workstation you turned the mouse on. When you opened a desk accessory, it was using a different workstation and considered the mouse to be turned off, but it was not according to your program. It took a few clicks to get it straightened out. The solution is simple—wait until your workstation is opened to do anything to the mouse.

Development Tools

Q: I cannot get MADMAC to generate a listing of my assembly. What can I do to get a listing?

A: MADMAC is a one-pass assembler. Switch order on the command line is important. Command lines are processed from left to right in one pass and switches generally take effect when they are encountered. The recommended procedure is to specify all switches before the names of any input files. For example, the command line: mac foos.s -l bar.s will produce a listing of bar.s and a symbol table, but NOT a listing of foo.s.

Atari Markets

Continued from page 1

makes them especially suited for desktop publishing, simulation, macro modelling, robotics, emulations, financial modelling, music emulation, speech synthesis, and general office applications. High specification video output provides for image processing, computer-aided design, and TV and film work.

New Atari hardware—such as the CD-ROM and floating point coprocessor— cannot be utilized until Atari developers produce the software that drives it.

On the Outside Looking In

Comments by dealers, sales reps, and real-world users can provide insight into markets that require further development, or the potential for ground-level opportunities. The following quotes were made to an Atari representative last month.

"Developers should definitely be focusing on development of strong educational programs like the Arrakis series. Educational programs should come with workbooks, text books, things the students can transcribe back to the teacher for evaluation, and to do homework if they have no computer at home."

—Atari district sales manager (former teacher)

"There appears to be a need for statistics programs, such as statistical analysis for psychology, etc."

—salesperson, university bookstore

"Atari needs to take advantage of its special capabilities: graphics, speech, music, the mouse, etc.

"Two real teaching needs must be addressed by software developers. The first is good software for young children, preschool and kindergarten. Much of the available software is written by people who don't have kids, or who have never taught. The second is software that teaches higher level thinking skills, especially for students in high school and college."

—high school teacher who works with Atari STs

Working Together

The Developers Conference scheduled for later this fall, as well as the Comdex show in November, will provide an opportunity for developers to closely examine market growth and needs in the Atari environment.

Working in conjunction with executive and marketing management at Atari, developers can make an investment in the future. It is only with the support of its developers that Atari can attain these goals.

Atari Policy and Software Piracy

by Dennis Hawker
Corporate Security Director

As discussed in our last edition, there are three principal forms of activity that constitute software piracy: illegal mass reproduction, electronic distribution through bulletin boards, and casual copying. In follow-up, I would like to keep you, our software developers, as well as end-users, informed of Atari's on-going efforts to curb such activity.

First, Atari will make an all-out effort to educate the public as to the wrongfulness and consequences that result from pirating software. Secondly, Atari has adopted an aggressive policy to investigate and prosecute those persons responsible. All investigation will be submitted to local and federal attorneys for criminal prosecution. Our legal staff at Atari will prosecute those responsible civilly. The bottom line is simple: Atari will not fall victim to software piracy. My department is responsible to investigate all matters relating to software piracy and I urge anyone with information to contact me directly: Dennis Hawker, Corporate Security Director, Atari Corporation, 1196 Borregas Avenue, Sunnyvale, CA 94086-3427, (408) 745-4319.

I will use this forum in the future to update you on concluded software piracy investigations.

New Online

In data library 7 (for registered Atari Developers only) in the Atari Developers SIG on Compuserve, the following files are new this month:
QA10.DOC--last months questions, GDOS.DOC--how to use GDOS with your application. In Library 4 in the Atari Developer's Roundtable on GEnie, QA10.DOC is file #52 and GDOS.DOC is File #53.
Correction: In the BUTTON.ARC program uploaded to Compuserve and GENIE there is an error which appears in the C source file button.c. The declaration:

```
long BUT_ADDR
```

should be:

```
extern long BUT_ADDR
```

MEGAFILE 20

Atari's new 20-Mb hard drive, the MEGAFILE 20, is now available. The MEGAFILE 20 is compatible with the entire line of MEGA and ST computers, and it is conveniently packaged to fit under a MEGA. For pricing and more information, please contact Cindy Claveran. The MEGAFILE 20 replaces the SH204 hard drive, which has been discontinued.

Events Calendar

As a service to Atari users, dealers, and developers, a calendar of local and national computer events will be compiled. Announcements of such events should be sent as early as possible to Elizabeth Shook at Atari Corporation, 1196 Borregas Avenue, Sunnyvale, CA 94086. Or call Cindy Claveran at (408) 745-2568. Be sure to always include the name and number of a contact person to call for more information.

AUGUST

20-21 : Ohio. Computerfest with participation by MVACE and others, Hara Arena, Dayton, OH. Bruce Hansford, chairman (513) 439-1993.

18-21: Wisconsin. National games show, GenCon, with participation by MilAtari (Milwaukee). Bruce Welsch, president, (414) 774-5253.

SEPTEMBER

2-4: West Germany. Atari Messe, Dusseldorf. Contact Atari Deutschland, Frankfurter Strasses 89-91, 6096 Raunheim, West Germany, phone (49) 6142-2090.

10: Georgia. Computer show, Houston Mall, Warner Robins, GA. Atari and IBM compatibles. Contact Peter Miller, Middle Georgia Atari UG, (912) 922-5666.

15-17: California. Seybold Desktop Publishing Exposition, Santa Clara Convention Center, Santa Clara, California. For more information, call Seybold Seminars, (213) 457-5850.

16-17: California. Southern California Atari Computer Faire, Version 3.0., Glendale Civic Auditorium, Glendale, CA. Produced by ACENET, a group of 22 computer clubs, including HACKS. Space for 80 exhibitors, over 5000 attendees expected. John King Tarpinian, president, (818) 760-1831.

OCTOBER

1-2: Washington, D.C. WAACE AtariFest, Fairfax High School, Fairfax Virginia. Gary Purinton, chairman, (703) 476-8391.

15-16: California. Atari Expo, San Jose Convention Center. Bob Barton, president, (408) .

NOVEMBER

14-18: Nevada. Comdex '88, Las Vegas. Atari has reserved the Gold Room. Write: Registration Dept., Comdex Fall, 300 First Avenue, Needham, MA 02194.

TOS 1.4 Development

by Ken Badertscher
Atari R&D Support

Beta testing of the latest release of TOS is nearly complete. The 5/18/88 Beta Test TOS was sent to Atari subsidiaries world-wide, and the response we received was very positive. Our thanks go to those whose System Problem Reports (SPRs) helped us to eliminate the final few problems with the new software. The SPR format, designed to simplify bug reports and enhancement requests, combined with the Product Tracking System (PTS) database, has helped the TOS developers in Sunnyvale a great deal. The PTS continues to meet its goal of keeping Atari aware of beta test findings and suggestions for new features.

Among the enhancements initiated as a direct result of the beta test are:

- Improved documentation of various OS functions
- Better redraws of the File Selector
- Improved Desktop window handling
- A more standard interface in Desktop dialogs
- Several improvements in new Desktop features
- Time/date stamp now preserved by file copy
- VDI escape functions made more robust

We have a Developer Release of TOS 1.4 available, so that you can have a head start taking advantage of the new features. The Developer Release will be a RAM-loaded version (just like the good old days!). It will come with a complete set of release notes describing changes and enhancements to TOS and how they might affect your software. To receive this developer release of TOS 1.4, please see the "Developer Kit Update" section.

Details of the PTS and how you can submit System Problem Reports will also be included in the Developer Release of TOS. SPRgen, a program for generating SPRs in the approved format, will be included on the TOS boot disk. SPRgen creates text files which are parsed and integrated directly into the PTS database. SPRs should be submitted electronically on CompuServe, BIX, GEnie, or the Atari Base Bulletin Board Systems. Details of how to submit problem reports will be included in the PTS documentation.

The Product Tracking System is our primary means of identifying and resolving problems with Atari hardware and software. With your help, we will be better able to document, track, and solve problems you are having with TOS and other Atari products. You can also use the PTS to suggest enhancements to Atari products and documentation. All System Problem Reports are reviewed by Atari engineers, and you can get feedback if required. Working with the PTS results in timely resolution of problems.

If you have questions about the Product Tracking System, you can send electronic mail to the PTS coordinator via Usenet: {portal,ames,imagen}!atari!pts, or contact me on BIX (kbad), GEnie (SYNERGIST), or CompuServe (71531,715).

Developer Kit Update

Registered developers are invited to apply for a set of revisions and additions to the ST Developer Kit. The complete set includes:

Developer Release of TOS 1.4
- floppy-based, RAM-resident
- Release Notes

SFP004 Developer Kit (MEGA Floating Point PCBA)
- routines and demos
- Developer's Manual

MadMac macro assembler and tools
- 4 bug fixes
- manual and release notes

'aln' linker
- several bug fixes and enhancements
- manual and release notes

'db' object-level debugger
- first release! (extensively used internally at Atari)
- manual

Please send a check for $20 to Cindy Claveran, Atari Corporation,1196 Borregas Avenue, Sunnyvale, CA 94086. The $20 covers our production and shipping costs for this set of updates.

Accessories, Pop-Ups, and Main Applications

by Richard Body

Introduction

The trend to larger random-access memories in personal computers has lead to increasing numbers of complete single-user-dedicated environments. Many PC users now expect to find (or create) a special set of tools that work together harmoniously and can perform related tasks without disruption of the main task—the task being directed towards a primary goal, such as publishing a multi-page document.

The Macintosh resorts to "Switcher" type programs. Accessories for the Macintosh must not use more than 32K RAM. In the MS-DOS world there are no standards—although Borland's "SideKick Plus" provides a de facto gap-filler. OS/2 is a huge response to this lack of standards although it also appears to have a mammoth's appetite for resources.

For the Atari ST, of course, we already have GEM's Application Manager. In this article I will discuss mechanisms already supplied for the Atari ST by DRI's

Continued on page 5

Accessories

Continued from page 4

GEM, and I will propose a partial set of standards for message-passing amont accessories and applications.

I will also present two accessories conforming to these standards. One accessory checks spellings and looks up synonyms. The second allows other accessories to be opened with hot-keys.

Desk Accessories

According to DRI "A desk accessory is an application that does not take over the entire screen."1 Desk accessories are loaded at system-boot and never terminate.

We already have, or can soon expect to see, many good examples:

1. keyboard macro expanders and enhancers;
2. system measures/controls;
3. alarms, clocks, calendars;
4. memo, notepad editors;
5. spell-checkers, synonym-finders, idea-outliners;
6. debugging aids, program profilers, software metrics;
7. file-squeezers, file-locaters;
8. screen dimmers, output device setups;
9. MIDI, modem and print spoolers.

Limitations: Room for at most 6 accessory titles registered under "Desk" Menu. Room for creation of at most 4 windows owned by accessories.

HandShaking on a Loan of Resources

Among even a limited number of concurrent tasks, cooperation is probably best implemented by a systematic discipline of sharing data and operations. Some of the discipline is provided most naturally by the AES Application Manager and its functions appl_find, appl_writ and appl_read.

Whenever two tasks share data, each must exercise restraint while manipulating the other's information. If an accessory changes the main application's data without its permission or knowledge, catastrophe can result. Remember that either accessory or main application can become active at any AES call. Even reading data without the donor's permission can be dangerous—the data may become rapidly out-of-date. We will define non-catastrophic borrowing of data in terms of "primitive" operations. A primitive operation on some datum must guarantee the integrity of the datum for the entire duration of the sharing. The data-structure plus all of its primitive operations is known as a "monitor."2

When the main application terminates, all open accessories are automatically closed, so in most cases, it is the main application which is the foremost guardian of the (user's) data. For this reason I shall focus mainly on situations such as when an accessory requests permission to view and perhaps modify some (text) data structure of the main application. I will call the recommended communications protocol a "handshake" consisting of three transactions, called the request, the reply and the release.

The handshake begins when the accessory is open and wishes permission to use some data structure of the main application. The accessory sends (with appl_writ) a pre-defined message to the main application requesting the main application's help in accessing the data-structure.

The main application will make a decision about whether it can safely share the requested data. If it decides not, it simply ignores the message and, two seconds later the accessory has to fend for itself as best it can (probably by complaining to the user). To reach a decision about sharing data, a main application must be able to guarantee that the data in question will not be changed in some conflicting way so long as it is allowing the accessory access to the data. If the main application decides it is safe to grant access it replies, telling the accessory how it may safely access the data in terms of primitive operations.

In return , the accessory must relinquish the data at the very first possible moment—by sending a message to the main application saying, in effect, "Thanks, no more access needed for now." Until the accessory so replies, the main application will be severely limited in what it can do, because it must continue to keep unchanged the data it is sharing.

Although GEM provides for arbitrarily long messages, the standard pre-defined messages which GEM uses are all 16 bytes long. Tom Hudson3 makes the suggestion that replies to requests should be numbered hexadecimal 80 beyond the request's number. I believe this is a good suggestion since it can resolve possible synchronization problems. Patrick Bass' article4, in the same magazine, develops accessory interfaces for Degas Elite graphics in the same spirit that I will use for a text-processing accessory. In the case of Degas Elite, the structure shared is a screen-buffer. In the example that follows, the structure will be a stream of text.

An Example : Handshaking on a Secure Word Stream

An accessory provides auxiliary text-processing according to the functional definitions given below5: the accessory will check the spelling of a word against its own dictionary. It can also shade the meaning of a word with its own thesaurus.

Continued on page 6

Accessories

Continued from page 5

```
/**** <FIGURE 1> HandShake (Accessory's Point-of-View) ****/
#include <define.h>
#include <gemdefs.h>
extern int gl_apid;

#define ENQ 5
#define ACK 6
#define REPLY 0x80
#define WORD_STREAM 0x50     /* data-type and access-method*/
#define CURRENT_WD 0x10      /* data-instance */
#define MAIN_ID 0/*CAVEAT NOT DOCUMENTED*/

typedef char* StringFcn(); /*function returning string */
typedef int * IntFcn();

StringFcn* Getword;
IntFcn   * Changeto;

HandShake(data_shr)
int data_shr;
{    int message[8];
     message[0]=ENQ<<8|2;
     message[1]=gl_apid;
     message[2]=0;
     message[3]=data_shr;    /*WORD_STREAM<<8|CURRENT_WD;*/
     Getword=NIL;
     Changeto=NIL;
     if(!appl_writ(MAIN_ID,0,message))
               return(FALSE); /*main program is no GEM*/
     for(;;)
     {    int evnt=evnt_multi (MU_TIMER|MU_MESAG,
                         0,0,0,0,0,0,0,0,0,0,0,0,0,
                         message,
                         2000,0,
                         &evnt,&evnt,&evnt,&evnt,&evnt,&evnt);
          if(evnt&MU_TIMER)
               return FALSE; /*timed-out*/
          if(  evnt&MU_MESAG
            && message[0]==(ENQ+REPLY)<<8|2
            && message[1]== MAIN_ID
            && message[3]== data_shr
            )
          {    Getword= ((StringFcn**)(message+4))[0];
               Changeto= ((IntFcn**)(message+6))[0];
               return(TRUE);
          }
     }
}
Release(data_shr)
int data_shr;
{    int message[8];
     message[0]=ACK<<8|2;
     message[1]=gl_apid;
     message[2]=0;
     message[3]=data_shr;
     Getword=NIL;                /*primitives no longer valid*/
     Changeto=NIL;
     if(!appl_writ(MAIN_ID,0,message))
     {    Assert("Main application no longer functioning!");
          Assert("Data just processed may be corrupted!");
     }
}
Check_spelling_of_current_word_of_main_application()
{    char *current_word;
     if( HandShake( WORD_STREAM<<8 | CURRENT_WD )
       &&(    current_word = (*Getword)()  )
       )
     {    char  corrected_word[132];
          if(spelt_wrong(current_word,corrected_word))
               if (! (*Changeto)(corrected_word) )
                 Assert("Main program won't accept change");
          Release(WORD_STREAM<<8|CURRENT_WD);
     }else Assert("Can't get current word from main pgm.");
}
Assert(s)
char*s;
{    Cconws(s);Cconws("\n");}
/************************* END of FIGURE 1 ******************/
```

The Accessory's Request

To access text in a word processing program, the accessory sends one of the following 16-byte messages to the main application.

byte[0] = ENQuire = 0x05 (Ascii control code-ask for resources)

byte[1] = N_Primitives = 0x02 (this access method requires only two primitive operations- see below, at main-applications reply.)

word[1] = accessory_id = sending accessory was given this by AES.

word[2] = 0 = number of bytes in message in excess of standard 16.

byte[6] = general classification of data structure = 0x10 (Word_stream)

byte[7] = unique description of data structure = (one of)
  0x10 current word
  0x20 current line
  0x30 block of words most recently selected by user
  0x40 top-most window
  0x50 current text_buffer
  0x60 current filewords[4-7] unused.

The accessory will wait two seconds for some message in response.

The Main Application's Reply

As noted above, if the main simply ignores the accessory's request and sends no message in reply, the accessory will consider this a "no" response, and make alternate provisions if possible. However if the main application wishes to provide access to the structure specified in bytes 6-7 of the message, it will provide a means of (securely) accessing the structure. In the case of Word_stream, two primitive functions, written in C or the equivalent,are necessary:

char* Get_Ascii_translation();*/for first ungot word instream,starting at the beginning of the structure; return NIL pointer (0L) if stream exhausted*/

int Replacement_with(Ascii_string)/* change last-gotten word of the stream with internal representation of parameter; return value 0 if replacement not successful.*/

It is the responsibility of the provider (the main application program) to ensure that none of its other actions interfere with these primitives during the duration of the access. The access begins with the sending of the reply in the following format:

byte[0] = ENQ + 0x80 = Reply to request.

bytes[1] same as corresponding byte of the request.

word[1] = main_application's i.d. (issued by AES)

word[2] = 0 = length of this message in excess of standard 16 bytes (nonzero if more than two primitives required for some access method)

long[3] = pointer to Get_Ascii_Translation();

long[4] = pointer to Replacement_with();

Release from the Accessory

The access (and the handshake) end when the main application receives the release from the accessory.

byte[0] = ACK = 0x06 = Ascii control character ("Thank you")

byte[1] = same as corresponding byte of the request.

byte[2] = accessory's id (same as in request).

word[3-7] same as in reply.

Continued on page 7

Accessories

Continued from page 6

A strict definition of a possible word is probably necessary here. A word is a sequence of alphanumeric letters, separated by delimiters. Delimiters include blanks and special characters, such as commas, periods, colons, hyphens and so on. However apostrophes and digits are considered to be letters.[6] The Ascii strings mentioned above must be in the range 32-126 decimal and must be NULL-terminated.

Recovery from Catastrophe

Although the above protocol is designed to secure the shared data structure, it is in the nature of mortal designs that there's no perfect security. Accordingly designers of accessories should take extreme precautions to release their programs from unneeded handshakes even in the most extreme circumstances. Designers of main applications should provide a "bail-out" feature in each of the primitives supplied, so that if the unexpected happens, the primitives can refuse to provide further access (stream exhausted, replacement not possible etc.).

```
/*************** FIGURE 2 Primitives *******************/
#include <define.h>
#include <gemdefs.h>
extern int gl_apid;

#define ENQ 5
#define ACK 6
#define REPLY 0x80
#define WORD_STREAM 0x50     /* data-type and access-method*/
#define CURR_WD  0x10        /* some data-instances*/
#define CURR_LN  0x20
#define CURR_BLK 0x30
#define TOP_WIND 0x40
#define CURR_BUF 0x50
#define CURR_FIL 0x60

typedef char* StringFcn();   /*function returning string */
typedef int * IntFcn();

typedef struct               /* format of request-to-share-word-stream*/
{    char event, n_primitives;
     short requester,extra_length;
     char rsrc_type;
     char rsrc_instance;
     StringFcn *primtv1;
     IntFcn *primtv2;
}Wds_Rq;

extern   /*very external...please ignore*/
user_has_already_selected_a_word,
in_some_vast_reorganization_of_stream,
we_like_sharing_data_with(),
n_open_windows;
extern int how_many_words_shared;
extern char copy_of_current_word[];
extern hypernode[],translate_();

rcv_message(msg)/*application's general message_handler*/
int *msg;
{    switch ( msg[0] )
     {case WM_TOPPED: /* and so forth*/

          case MN_SELECTED: /* and so forth*/
               break;
      default:if( msg[0]>>8==ENQ
              &&msg[3]>>8==WORD_STREAM
              )
               share_word_stream((Wds_Rq*)msg);
     }
}
share_word_stream(rq)   /*word_stream_request_handler*/
Wds_Rq* rq;
{    switch(rq->rsrc_instance)
     {case CURR_WD:
          if(  user_has_already_selected_a_word
            && !in_some_vast_reorganization_of_stream
            && we_like_sharing_data_with(rq->requester)
            )
               share_current_wd(rq);
          else return FALSE;

      case CURR_LN: return FALSE;
          /*we're hypertext-,not line-oriented*/
      case TOP_WIND:if(n_open_windows>0)
                    share_top_window(rq);
               else return FALSE;
      default: return FALSE; /*unknown resource-instance*/
     }
}
share_current_wd(rq)
Wds_Rq*rq;
{    StringFcn one_time_only;
     IntFcn ascii_to_current_word;
     Wds_Rq release;
     int accessry= rq->requester;
     how_many_words_shared=1;
     rq->primtv1=one_time_only;
     rq->primtv2=ascii_to_current_word;
     rq->requester=gl_apid;
     rq->event |=REPLY;
     if(!appl_writ(accessry,0,rq) )
     {    Assert("Accessory made bogus request");
          return;
     }
     do   evnt_mesag(&release);
     while(    release->event    != ACK
          || release->requester!= accessry
          || release->rsrc_type!= WORD_STREAM
          || release->rsrc_instance!=CURR_WD
          );
}/*this is the simplest way of ensuring nothing harmful is done during hand-
shake--do nothing at all!!! There are more sophisticated methods*/
/***********Imaginary examples of primitives ***********/
char* one_time_only()
{    if(how_many_words_shared-->0)
     {    translate_from_hypertext
          ( hypernode[0],copy_of_current_word);
          return copy_of_current_word;
     }else return NIL;
}
int   ascii_to_current_word(ascii_string)
char * ascii_string;
{    return translate_to_hypertext
          (ascii_string,hypernode[0]);
}
/*****************End Of Figure 2 ************************/
```

PopUps and Hot-Keys

Macintosh accessories can often be opened by pressing a combination of "hot-keys," so that the user does not have to lift his hands off the keyboard. Similarly MS-DOS users can activate "transient and stay-resident" programs with "pop-up" keys. If you install an accessory POPUPS before installing your own accessory, you can register hot-keys with POPUPS. POPUPS will monitor the keyboard, and open the accessory when you hit the registered key-combination.

We can express the interaction between the user's accessory and popup as an unusual form of handshake: the user accessory requests a service of PopUp, namely to translate a key-combination[7] into a message AC_OPEN. It is unusual in that the data shared, the accessory's AES i.d., is guaranteed to be valid for the lifetime of the accessory, which is eternal! The handshake likewise need never be released.

```
/************** FIGURE 3 Popup Accessory ****************/
/*PopUp is an accessory not appearing under Desk-Accessories.
     1)It registers the "hot keys" of other accessories.
     2)It watches the keyboard for pressing of any registered hot keys.
/*How OurAccessory registers its hotkey(s) with PopUp:*/
/* but first a few definitions*/
#include <gemdefs.h>

#define SERVICE 0
#define POPKEY 0x05
#define ENQ 5
#define AESkeycombo 0x1400   /*Alt-T see [7 ]
struct popmsg
{    char event, n_primitives;
     int apid,msglen;
     char type,instance;
     int menuid;
     long keyscan;int ext;
}PopKey;
/* Now, while initializing Our Accessory....*/
     extern int gl_apid;
     int gl_menuid=menu_register(gl_apid,"  OurAccessory");
     int pop_id=appl_find("POPUP   ");
     if(pop_id>=0)
     {    PopKey.event=ENQ;
          PopKey.n_primitves=0;
          PopKey.apid=gl_apid;
          PopKey.msglen=0;
          PopKey.type=SERVICE
          PopKey.instance=POPKEY
          PopKey.menuid=gl_menuid;
          PopKey.keyscan= AESkeycombo
          appl_writ(pop_id,16,PopKey);
          event=evnt_multi(MU_TIMER|MU_MESAG,
                    0,0,0,0,0,0,0,0,0,0,0,0,0,
                    &PopKey,
                    2000,0,
                    &evnt,&evnt,&evnt,&evnt,&evnt,&evnt);
          if(evnt&MU_TIMER)
               Assert("PopUp accessory not available");
          if(  evnt&MU_MESAG
            && message.event==POPKEY+REPLY
            && message.apid==accsry_id
            )
     }else     Assert("No PopUp accessory found..");
```

Continued on page 8

Accessories

Continued from page 7

The POPUPS accessory and its source files are uploaded on GEnie's Atari Developers' Forum.

Conclusion

It remains problematic whether most users of personal computers need or want the overhead and complication of full unlimited multi-tasking systems. Accessories are a useful compromise for building computing environments dedicated to one primary goal, such as desktop publishing, graphics design or music composition. Developers will find a rewarding set of challenges in building tools for such environments.

1 GEM Prog. Guide, Vol. 2: AES: 1.5.2.1
2 Per Brinch Hansen, "The Architecture of Concurrent Programs," Prentice Hall, 1977, pp. 19-22, 52-54.
3 Tom Hudson, *START* Summer 1987, Vol. 2, No. 1 "Plumbing GEM's Mysteries" pp. 43-47.
4 Patrick Bass, *START*, Summer 1987, Vol. 2, No. 1 "A Super Toolkit For DEGAS Elite" pp. 23-29.
5 Julius Oklamcak, "Application to Spelling Checker and Thesaurus Interface Specification" Atari Canada Corp., October 1987
6 James L.Peterson, "Lecture Notes in Computer Science #9," Springer-Verlag, 1980, p. 3.
7 Sheldon Leemon, "Technical Reference Guide Atari ST Volume 2" *Compute!* Publications Inc., 1987, Appendix B7.

Best of the Boards

by Ken Badertscher
Atari R&D Support

Electronic Bulletin Board Systems (BBSs) and online services are a great source of information and helpful tips about all facets of Atari development. After sifting through hundreds of kilobytes of text files captured from three of the major information networks, CompuServe, GEnie, and BIX, I came up with a representative sample of the kind of messages you can see every day online. Each of the networks has its own personality, but they all have something in common: helpful users and sysops. Chances are pretty good that you can pose an electronic question on just about anything, and useful feedback will be waiting for you next time you log in.

The CompuServe AtariDEV Forum is a direct line to Atari developer support. Topics in the forum include open language and hardware sections along with two areas for registered developers only. The discussions represented here range from evaluating a C development system to talking to the 68000 on the Mega bus.

GEnie is a popular information service because it is less expensive than many of the others. The ST RoundTable on GEnie is one of the most popular. GEnie also has a RoundTable just for registered Atari developers, also called AtariDEV. In the messages here, a malloc() question results in a description of the public domain dLibs C library, and Neil Harris opens discussion on the upcoming developers conference. BIX (the Byte Information eXchange) has a wide variety of talent online representing fields from Artificial Intelligence to Zoology. Atari has four conferences on BIX: atari.st, ataricorp, atari (for 8-bit computers) and the developer conference, ataridev. Excerpts include using BIOS calls with MIDI, info on Pexec 4 and 5, and GEMDOS standard handles.

From the CompuServe AtariDEV Forum:

AZTEC C

Fm: Corey Cole 76224,66

I haven't seen the Aztec source debugger yet (just saw an ad for it yesterday), but have been using the rest of the package heavily for a month or so now, and I like it. I chose Aztec (even though I've been using Megamax for another project for some time, and am very impressed with it as well) because I am developing for the Mac and Amiga as well, and Manx has essentially identical compilers for all three machines. Also, I like having a separate assembler, as the project I'm converting (from the PC) is about 1/3 assembly language, and I didn't like the idea of doing so much assembly code with an in-line assembler.

I would say that compilation speed with Aztec is a little slower than with Megamax, but my off-the-cuff impression is that Aztec's code generation is superior to Megamax's (seems to have better optimization, makes good use of the 68000 instruction set).

The documentation on how to create Alcyon-compatible symbols (for use with SID) is incorrect, but the feature does work. The only other negative I have is that the compiler gets unhappy about CTRL-Z end-of-file characters (doesn't kill it, but puts out lots of annoying warning messages). On the whole, it's a very solid and well-documented package, comparable to Megamax, but with the mentioned advantages (portability and separate assembler.

General Information

Fm: Winston M. Llamas 75176,2260

Thanks so much for the information. I have been trying to compile an expert system shell on the ST for a while now, but Megamax and Mark Williams always choked on it—Megamax because it isn't "ANSI" enough, and MWC kept generating an internal compiler error (since the program worked on my other system I never bothered modifying the code—lazy I guess). I've used Aztec on the PC and on the Mac and I like it—it

Continued on page 9

Best of the Boards

Continued from page 8

looks like they keep the quality up across the boards, so to speak. And it's a pretty good deal with their current special.

Thanks again.

PTERM

Fm: SYSOP*Keith Ledbetter 76701,124

Ok, guys...here's another one. I have an application I'm writing that traps all BIOS calls, and under certain circumstances I need for my BIOS routines to blow the currently running program "out of the water" (grin). What's the best way to do this? Can I just execute a PTERM call from within my trap13 handler. If not, what could I do to cause a fatal error just on the currently executing process? Keith

Fm: John Feagans (Atari) 70007,1072

If you can do a pterm from trap #13 and survive, you would be extremely lucky... Here is the way I would do it: Monitor trap #1 as well. Set a global flag in the trap #13 that you want to do a pterm. The next time the application does a trap #1, change the function code to pterm and bye bye.

Fm: SYSOP*Keith Ledbetter 76701,124

Hmmm... that sounds like a good idea. It's probably the "cleanest" way to do it.
Thanks! Keith

Fm: Marc A. Pelletier 76340,3107

Actually, if you don't mind the bombs on the screen, just execute that bit of code:

```
        move.l   $8,a0
        jmp      (a0)
```

That will simulate an address error, put two bombs on the screen, and quietly return control to the calling program. Of course, you must be in Supervisor mode to do that, but then again, you are if you are trapping bios/xbios calls. Marc.

Fm: SYSOP*Keith Ledbetter 76701,124

Thanks, Marc. The bombs don't bother me at all (the SysOp's may be a different story, but...(grin)Keith

From the AtariDEV RoundTable on GEnie:

Sub: Use of Malloc()
Usage of Malloc(-1) to give you all available memory

J.ROBINSON12

Is there an amount of memory reserved that Malloc can't get to? I have 37,536 bytes free listed when using the Intram accessory with DBMaster and my own program that can't be used. Is it just not possible to use this memory? If so, does this amount of unusable memory remain the same with different memory configurations?

JLS *[John Stanley]*

J.Robinson12, you might want to use the dLibs malloc routines (and dstart startup module) to avoid the space limitation problems in the Alcyon libraries. The dLibs memory allocation system uses Malloc (GEMDOS call) to take a huge block of memory (programmer defineable size) and then parses it out in smaller chunks via the C malloc() call. This means that -all- system memory is available for your allocations, you don't run into the 20 Malloc problem, you don't have to worry how small you make your runtime stack (because your malloc() calls don't care), and you can still run Pexec'ed programs from within your program and they'll use almost all non-malloc'ed memory to run in.

I have several interacting programs (editor, shell, etc) that all use the dLibs functions. I can pull in a monster text file almost as large as free memory, or I can pull a couple of smaller files, run Flash from inside the editor,drop out of Flash, and still be able to pull in a large text file. Frankly, I can't imagine why anyone would want to stick with the very limited Alcyon memory management when dLibs is available as a public-domain library that's so much more bug free than the Alcyon libraries...

dLibs is available in the main ST RT in GEnie. It's fully compatible with the Alcyon compiler and source code is provided free of charge. I know I sound like an ad, but I also know of several people who have spent months fighting to get the Alcyon libraries to work. They then tried relinking with dLibs and major sections of code suddenly started to work...

Sub: Developer Conference

Atari is planning a major conference for developers, to be held in Sunnyvale immediately following Comdex.

NHARRIS *[Neil, Atari U.S.]*

We have a meeting scheduled next week to discuss the details of the first ever USA conference for Atari developers. Now is the time for some input. What sorts of activities do you think should be included?

J.OKLAMCAK *[Julius, Atari Canada]*

Neil, I can see the conference taking two directions. One for marketing folks and the other for programmer folks. First an intro of where Atari is going in terms of marketing and products. Then the two groups could be split up into different sets of seminars. I would like to see the Atari Engineers on hand to answer questions, programming seminars (how-to's, why-for's, etc.) For marketing folks you could have info on the different avenues of selling products, selling into other English speaking countries, and even to foreign language countries. (Or is this going to be an international developers conference?) Maybe even a revised set of docs could be ready by then (grin). Possibly set forth some standards for applications to follow...

Continued on page 10

Best of the Boards

Continued from page 9

R.WARSHAW at 19:55 PDT

A good idea. I'm sure many developers would welcome a chance to discuss Atari's present and future plans. Needn't be a shouting match either.

Ray Warshaw

JR.WILSON [WPCorp] at 14:31 EDT

Programming for upward compatibility with future ST designs might be nice.

N.WEINRESS at 22:21 PDT

I echo that last statement. Everybody knows that Atari must replace the ST with a more capable machine. Upward compatibilty, or the lack of it, is going to be a vital question.

C.GREENE3 at 13:37 EDT

Yes, upward compatibility would be great. How about scheduling part of the conference to deal with this? Possibly a discussion on OS changes that might be offered to allow programmers a chance to reduce the amount of direct hardware stuff that is done... Chris

From ataridev and ataricorp on BIX:

MIDI

ataridev/internals #50, from ggf *[Gary Frederick]*

We are playing with the MIDI port and would like to have some info on what the MIDI port really does.

Does TOS flow control work with the MIDI port?

If I read/ write the MIDI buffer directly, will I get extra control characters or will it just have what I put in?

I did a few tests of bcon* and the MIDI seems to be sending at around 1/3 of the max possible. Do the bcon* calls have some overhead that TOS uses?

Jefferson Software will be modifying the zmodem programs from case to use the MIDI port. Has anyone other than Fred Brooks with mx2 done anything with the MIDI port and ST used as a file server?

ataridev/internals #51, from john.r.strohm

The MIDI 1.0 Specification does not define any flow control protocol. Some of the instrument manufacturers (Roland comes to mind) have defined flow control as part of their System Exclusive stuff.

When reading the MIDI buffer, you will get exactly what was received from the other end, no more no less. Be aware that some instruments DO send active sensing bytes every so often. (Yamaha DX7, the original, sends every 80 msec.)

The Atari ST MIDI receive data interrupt handler has a defect. It cannot use more than 32K worth of buffer. If you are going to receive more than 32K bytes at a whack, you MUST empty the buffer on the fly. (The Oberheim Matrix-6 synthesizer Sys Ex bulk voice dump sends 34K bytes. Guess how I discovered the problem in the interrupt handler?) Beyond these flakes, there is not much to using the MIDI port. The BIOS and XBIOS calls work well enough.

Good luck!

PEXEC

ataridev/internals #62, from ggf

Can someone who is wise beyond belief tell me about pexec 4 and 5?

I can get one pexec 4/ 5 combination to work, but it—uh—fails if I pexec 4 several programs then try to run them via pexec 5.

ataridev/internals #64, from tempel *[Thomas Tempelmann]*

In the Megamax Modula-2, I've used that Pexec 4 & 5 to load programs resident and executed them from RAM. There are two problems:

First, many programs (often written in C) use the DATA segment and change it. That means, if you load a program with Pexec and run it, that program changes its own pre-initialized data. Then, if you run it again with the changed data, it'll get confused. To fix this problem, you must save the data segment every time you exec the loaded program and restore it when it has returned. You can get the location and size of the data segment out of the base page.

The second problem is that the TOS developers made a mistake. Every time a process (prg) is called, the system needs some initialization, e.g. the file handles, redirection, and so on. Pexec does it, but at the wrong time. It does it when you load the program and not when you call it. So, the system gets confused when you call a process twice because its resources are only initialized when executing the first time. Then, after the termination of the first process call, it closes its resources. When calling the second time, the process uses the old, closed/invalid resources. Because of that, it could happen, that redirection (from the parent process) and current paths will not be active/found when doing in the second and further process calls. As we are using the program load/execute in our Modula-2 system nevertheless, I'd say that it doesn't make for big problems.

But I'll try to convince the current Atari programmers to fix this bug. As I've got a source (recompiled by hand) of the pexec function I'd say that it would not be so hard to change it. (I'm wondering why the developers know this bug but haven't fixed it yet).

—tempel

Continued on page 11

Best of the Boards

Continued from page 10

ataridev/internals #73, from kbad *[Ken Badertscher]*

Allan Pratt sez: Tempel's explanation of what Pexec 4 & 5 do and don't do (see message 64) is quite correct. He has obviously studied them well. However, fixing the way Pexec works is not as simple as it may seem. Not only that, but if it *is* fixed to do something different, people who have working code now may find that their code breaks. The evils of backward compatibility manifest themselves in many ways...

STANDARD HANDLES

ataricorp/tos.upgrades #90, from orc *[David Parsons]*

Another thing that would be very* nice in the new roms would be to follow Unix convention for the file handles. The old developers notes don't really* describe how the file handles work, but I've been told by (highly-placed sources" that handle 2 is pointed at aux: instead of being used as stderr. Not having a real stderr is a pain in the nether regions if you're trying to (a) redirect output but not error output or visa-versa or (b) run the st via a terminal attached to the serial port.

ataricorp/tos.upgrades #91, from ggf

I asked a LONG time ago what the standard handles point at. There are 6. What are the official devices for ALL 6.

ataricorp/tos.upgrades #92, from kbad

GEMDOS standard handles are as follows:

0 - stdin (CON:)
1 - stdout (CON:)
2 - (AUX:)
3 - (PRN:)
4,5 - not initialized (but inherited by child processes)

Because a lack of stderr is a documented GEMDOS bug, and some development systems may have worked their way around it in one way or another, it is not likely that handle 2 will be changed to stderr. Especially since there is probably some software out there that will break seriously if it expects 2 to be attached to the serial port.

Orc, why don't you roll your own UNIX(tm)-compatible stderr library code by fduping handle 2 and then fforcing it to whatever you want to be stderr? That is a perfectly legit way of getting a stderr that does what you need without causing problems, and it could be easily accomplished in your startup code. ttfn...

ken @ atari

STDERR

ataricorp/tos.upgrades #93, from orc

Why don't I do that in startup? Look at how Unix handles stderr. I could set things up so that my shell redirects handle 2 and makes it stderr, but then I'd have to require that all my programs are run from within my shell. (And that would make a lot* of unhappy STadel sysops—Gulam appears to be the shell of choice for the STadel sysops that use a shell.)

I want a stderr so that I can redirect it. If the startup code does dupping on stderr, stderr can't be redirected (my startup code, by the way, does not* do redirection itself—that is left, as it should be, to the shell.)

Has anybody done any testing to see whether people use the (undocumented) official handle 2? Keeping it as is is very silly—obviously someone was thinking MS-DOS when GEM was written, 'cause 0-5 are kept around, so why not try an experiment?

ataricorp/tos.upgrades #94, from ggf

If 4 and 5 are not initialized and passed to the child process, we can use them to redirect io to the child process. Is it official they are not initialized or is that just the current state of TOS?

ataricorp/tos.upgrades #95, from kbad

The current state is "not initialized." In the new GEMDOS, handles 4 and 5 point to the console. "Officially" I would say 4 and 5 are not initialized, since that's how it has been in TOS up to now, and we must be backward compatible. Yup, you could use handles 4 and 5 for special cases for child processes, since they're passed along in a Pexec. GEMDOS has no stderr. In a perfect world, GEMDOS would use handle 2 as stderr, or at least have a documented stderr. Alas, in the real world there are bugs and shortcomings in TOS.

Atari Corporation
1196 Borregas Avenue
Sunnyvale, CA 94086
Attn: Developer Support

Atari Developer Application for COMDEX Booth Space

COMDEX 1988 is upon us! The Fall show will be held in Las Vegas, November 14-18. Atari has arranged to use the Gold Room, a 6,600 square-foot hall off the main exposition floor. Atari has already begun formulating plans for its biggest and most exciting COMDEX show ever. And Atari wants YOU to be a part of it!

Featured at the show will be Atari's MEGA line of personal computers as well as new products. Atari will be emphasizing its ability to deliver solutions to the education, MIDI, desktop publishing, and graphics/animation markets. COMDEX is a chance for Atari and its developers to show we mean business!

Developers wishing to participate in Atari's booth at COMDEX must complete the following application. Applications must be submitted by September 1. Applications will be reviewed and qualifying developers notified by September 15. There is no fee for an Atari booth space if selected to participate. Call Joe Ferrari or Sig Hartmann if you have any questions about COMDEX or other shows. Accomodations in Las Vegas must be obtained by developers independently.

Atari would like to produce a booth directory of developers and products for the show. Please be sure to give clear, concise descriptions of each product and include pricing information.

Company ______________________________

Address ______________________ City/State/Zip ______________________

Company Contact ______________________ Title ______________________

Phone ______________________________

Customer Contact ______________________________

Phone ______________________________

Products to be shown (name and description): ______________________________

____

Exhibit Description ______________________________

Target Market ______________________________

Product Release Date ______________________________

Pricing (please attach price list if available): ______________________________

____

Equipment Requirements (**Atari reserves the right to refuse to loan hardware to any developer**):

Quantity	*Product*	*Part Number*	*Purpose*

Estimated Space Requirements ______________________ Estimated Number of Developer Attendees ______

Please detach and mail this application by September 1 to:

Joe Ferrari, Atari Corporation, 1196 Borregas Avenue, Sunnyvale, CA 94086