— CHAPTER 1 —

INTRODUCTION TO ATARI PROGRAMMING

Atari Computer Hardware

The 260/520/1040 ST

The first Atari ST computers became available to the public in 1985. The new Atari models were the first 16-bit computers well-suited for use in the home. The availability of these computers signaled the end of the Atari 8-bit era of computers such as the 400, 800, 800XL, 1200XL, 1450XLD, 65XE, and 130XE computers.

The name ‘ST’ is derived from the capabilities of the Motorola 68000 processor upon which the original Atari line was based. The 68000 uses a **S**ixteen-bit data bus with a **T**hirty-two bit address bus.

16-bit computers introduced a new concept in computer technology called the operating system (OS). Atari’s operating system, **T**he **O**perating **S**ystem (**TOS**), was loaded from a boot disk originally, but is now almost always installed in ROM.

A primary subsystem of **TOS** is **GEM** (‘Graphics Environment Manager’), the graphical user interface used by Atari computers. **GEM**, which was developed by Digital Research, Inc., manages the graphic interface to applications and provides access to popular computing features with buzzwords such as windows, the mouse, menus, and the desktop.

GEM was originally designed for PC-compatible computers. PC-based **GEM**, however, is no longer completely compatible with Atari **GEM**. Only components of **GEM** relative to its use on the Atari will be covered in this guide. Some functions which were originally documented for Atari **GEM** yet never implemented have been included for completeness.

Other **TOS** subsystems include **GEMDOS**, the **BIOS**, and the **XBIOS**. These subsystems provide a hardware interface and management functions for the file system.

The original ST computers featured the following:

- Motorola 68000 32-bit processor running at 8MHz.
- Integrated GEM/TOS operating system.
- RAM memory storage of 256k, 512k, or 1 Mbyte (depending on model).
- Built-in MIDI, dual joystick, floppy drive, ACSI, serial, and parallel ports.
- Sophisticated DMA peripheral access.
- Yamaha 3-voice FM sound generator.
- External 128k cartridge port.
- Integrated video controller capable of generating (320x200x16), (640x200x4), and (640x400x2) video modes from as many as 512 colors.

Mega ST 2/4

Two years after the release of the original ST series Atari released the Mega ST series of computers. The Mega ST computers were shipped with **TOS** 1.02 and featured several new features as follows:

- BLiTTER chip (for faster graphics).
- Internal expansion bus.
- Separate keyboard and CPU.
- Either two or four megabytes of RAM.
- Peripheral co-processor slot (for 68881 coprocessor, etc.).

STacy

The STacy was released shortly after the Mega ST to provide a portable means of Atari computing. STacy computers were shipped with **TOS** 1.04. The STacy's design supplemented the basic features of an ST with the following:

- Integrated CPU/keyboard/carrying case.
- Monochrome LCD screen.
- Track ball for mouse control.
- Optional hard drive.

1040 STe

The 1040 STe, released in 1990, was designed to expand upon the capabilities of the 1040 ST. Many of the features added were geared towards entertainment and multimedia applications. The 1040 STe was shipped originally with **TOS** 1.06. The following features were added to those of the basic ST:

- Extended color palette to support up to 4096 colors.
- Support for horizontal and vertical fine scrolling.
- Video GENLOCK capability.
- Stereo 8-bit PCM sound.
- Two extra joystick ports with support for paddles and light pens.
- 256k Operating System in ROM.
- SIMM memory slots to upgrade memory to 4 Mb

Mega STe

Released in 1990, the Mega STe was designed to provide for more computing power than the 1040 STe and add several new hardware features. The Mega STe shipped with **TOS** 2.02, 2.05, or 2.06. It adds features to that of a 1040 STe as follows:

- Motorola 68000 32-bit processor running at 8MHz or 16MHz.

- Optional 68881 math coprocessor.
- One, two, or four megabytes of RAM memory.
- Optional internal hard drive.
- Modern case design with separate keyboard/CPU.
- Three serial ports.
- Localtalk compatible networking port.
- VME compatible expansion slot.

TT030

Also released in 1990, the TT030 computer was the first Atari computer workstation designed for high-end computer users. The TT030 workstation was shipped with **TOS** 3.01, 3.05, or 3.06. It adds the following features to that of the Mega STe:

- Motorola 68030 32-bit processor running at 32MHz with cache.
- Memory capacity of 32Mb with optional 'fast' RAM.
- Standard 68882 math coprocessor.
- Four serial ports.
- SCSI device port.
- Stereo RCA jacks for sound output.
- Extra video resolutions of (320x480x256), (640x480x16), and (1280x960x2).

ST Book

Designed to replace the STacy as the defacto portable ST computer, the ST Book brought the basic computing power of an ST to a lightweight notebook computer. This machine was only released in Europe and Atari only shipped a very small quantity. The ST Book was shipped with **TOS** 2.06. Minus the internal floppy drive, it supported features beyond that of a STacy as follows:

- Lightweight case design.
- Keyboard with integrated numeric keypad.
- Mouse 'vector' pad.
- Processor-direct expansion slot.
- External keypad port.
- Floppy drive connector.

Falcon030

The newest member of the Atari line, the Falcon030 is to become the new base model Atari system. The Falcon030 is currently shipping with **TOS** 4.04. While remaining backwardly-compatible, the Falcon030 adds many new features as follows:

- Integrated case and keyboard design.
- Motorola 68030 processor running at 16MHz with cache.
- Motorola 56001 DSP with 96k RAM.
- Standard configurations with 1, 4, or 14Mb RAM.
- Internal 2 ½” IDE hard drive optional.
- Video resolutions from 320x200 to 640x480 with a palette from 2 to 256 colors and 16-bit true color.
- Adaptable to Atari monitors, standard VGA monitors, and composite video.
- GENLOCK-ready design.
- Ports include parallel, serial, external floppy, SCSI-2, LAN, 4 joystick, MIDI in/out, microphone, headphone, and ST compatible cartridge port.
- Interior processor expansion port.
- Sound system includes standard Yamaha FM chip, connection matrix, and 8-track, 16-bit stereo record/playback.

Atari ‘Clone’ Computers

Atari ‘clone’ computers first became available in early 1994. These computers, while mostly software compatible with Atari-produced computers, contain hardware enhancements and modifications that may cause incompatibilities in software that relies on hardware access rather than the recommended method of using standardized OS calls.

The recent availability of these computers as well as enhanced graphics and peripheral boards emphasizes the value of programming using the OS whenever possible to allow software to be run on the widest variety of machine configurations.

Atari Computer Software

GEMDOS

GEMDOS consists of file system management routines that provide access to all of the basic devices supported by Atari computers. It bears resemblance to MS-DOS in its functions and opcode numbering while still maintaining some differences and advantages.

MultiTOS

MultiTOS is the first truly multi-tasking extension to **GEMDOS** supported by Atari. Based on **MiNT**, developed by Eric Smith, **MultiTOS** adds true pre-emptive multitasking, memory protection, and process control. Its methods of job control and interprocess communication will be familiar to UNIX users. With the ability to support loadable device drivers and file systems, **MultiTOS** provides a complete range of functions to complement **GEMDOS**. In its current incarnation, **MultiTOS** is an option and thus disk-based as opposed to burned in ROM.

BIOS

The ST **BIOS** (‘Basic Input/Output System’) comprises the lowest-level of device communication. **GEMDOS** uses the **BIOS** to accomplish many of its file system operations.

XBIOS

The **XBIOS** (‘eXtended Basic Input/Output System’) controls other hardware-specific features such as the floppy drive, video controller, DSP, MFP, and sound system.

Atari GEM

AES

The **AES** is responsible for window and menu control, messaging services, and object rendering and manipulation.

VDI

The **VDI** consists of a series of drivers which provide device-independent access to the display screen and external output devices such as printers and plotters through **GDOS**. All graphic primitive operations are accomplished with the **VDI**. The **AES**, for instance, uses the **VDI** to render its objects on screen.

GDOS

GDOS is a disk-loadable subsystem of the **VDI**. The term **GDOS** can refer to original **GDOS**, **FONTGDOS**, or **SpeedoGDOS**. It controls loadable device drivers and fonts. The original **GDOS** was limited to bitmap fonts and did not have the bezier capabilities of **FONTGDOS** or **SpeedoGDOS**.

FONTGDOS

FONTGDOS is essentially a newer, faster **GDOS** with bezier rendering functions present. **FONTGDOS** is otherwise completely backwardly compatible with **GDOS.**

SpeedoGDOS

SpeedoGDOS, named for the Speedo™ font format created by Bitstream, Inc., adds outline font rendering capability to the basic features of **GDOS**. **SpeedoGDOS** also includes a sophisticated caching system to promote the fastest rendering possible.

Two versions of outline **GDOS** exist. The original version (referred to as Font Scaling Module (**FSMGDOS**)), based on QMS/Imagen fonts, was never officially released. Nonetheless, a small number of users still use **FSMGDOS** and differences between them are noted.

LINE-A

LINE-A is a special set of routines that provide an assembly language interface to routines and variables belonging to the **VDI** and **XBIOS**. It is so named because instruction opcodes beginning with the hexadecimal number $A utilize a special microprocessor exception which point to the **LINE-A** routines in ROM.

LINE-A is the only operating system component that has become out of date and incompatible. Atari recommends that software developers avoid using **LINE-A** as it will be supported less and less as hardware advancements make its use more incompatible. **LINE-A** is documented briefly in this reference for completeness.

Desktop

The 'Desktop' is a independent **GEM** application burned into ROM. It facilitates program launching and file manipulation as well as providing a graphical shell for user-interaction.

XCONTROL

XCONTROL (Extensible Control Panel) is a desk accessory application that provides access to multiple modules called CPX's (Control Panel Extensions) which are used to control system configuration and other related functions. A special section in this reference discusses the creation of CPX's and the utility functions provided by the **XCONTROL** shell accessory.

Third-Party System Software

Geneva

Geneva is an alternative, **TOS**-compatible operating system developed by Gribnif Software. It functions mostly as an **AES** replacement although it supplements other areas of the OS to provide cooperative multitasking (as opposed to **MultiTOS**'s pre-emptive multitasking).

Programming for Geneva 1.0 is identical to programming for **GEM** with **AES** version 4.0. Geneva does not currently support **MiNT** extensions though Gribnif has announced plans to eliminate this incompatibility in a future version. You can detect Geneva by searching for the cookie 'Gnva' in the system cookie jar. Likewise, the presence of **MiNT** extensions can be determined by the 'MiNT' cookie.

Programmers should not rely specifically on the presence of these cookies to determine if the current OS variety supports multitasking. The **AES** *global* array contains values to help determine the possible number of concurrent processes and the **AES** version number. In addition, the **AES** call **appl_sysinfo()**, available as of **AES** 4.0, can be used to determine the presence of special **AES** features.

Geneva offers several system extensions not available under **MultiTOS**. Information on programming the Geneva OS is available in the commercial package and direct from Gribnif Software.

Programming Languages

'C'

'C' has become the default standard for Atari computer programming. Most reference books and materials illustrate OS functions using 'C' style bindings. This book is oriented towards 'C' without, hopefully, alienating developers who develop in other languages. Several different 'C' compilers exist in the Atari domain. All have their various features and quirks which make it necessary to be familiar enough with your implementation to modify the source contained in this reference appropriately.

All ‘C’ bindings in this book were created for use with Lattice ‘C’ by Hisoft, Inc.. They should be easily convertable to other major Atari ‘C’ compilers.

Luckily, most ‘C’ compilers agree with their function naming and in most cases you can simply call the function as listed. If you have an older version compiler you may need to add some bindings using the information provided in accordance with your compiler’s recommendations.

Assembly Language

For the convenience of assembly language programmers, all functions are listed with their opcode and related binding. In addition, a section provided in front of the function reference will explain the calling conventions for functions in that category.

All assembly listings in this book were created for use by the AS68 compiler included in the Atari developer’s kit.

BASIC

Depending on the type of BASIC you utilize, functions may be named identically or differently from what is listed in this book. It is recommended that you seek a BASIC compiler that gives you proper access to all of the functions of the machine or familiarize yourself with a more robust language.

Other Languages

Various other languages exist in the Atari domain. Pascal, Forth, ‘C++’, and others have implementations that are similar in design to ‘C’. You should refer to your language manual to properly utilize information found in this reference.

Conventions

Typesetting

The following table displays a list of typesetting conventions used in this book:

Style	Meaning
Normal Text	Standard body text.
BOLD TEXT	Bolded words include function names like **appl_init()** , ‘C’ macros, ‘#defined’ data types like **WORD**, and operating system components such as **GEM** and **TOS**.
Italicized Text	Italicized text is used to represent

	variable names like *handle*. In addition sections of this book like ***AES** Reference Manual* will be in capitalized italic text.
| Text between vertical bars |	Vertical bars imply the absolute value of the variable or expression within. For instance: | -2 | == | 2 |
(Number 1, Number 2)	Two numbers contained within parentheses and separated by a comma indicate a coordinate point X followed by Y. For instance, (100, 100).
Number1 ^ Number 2	2 ^ 8 is the same as 2^8 or 2 to the power of 8.
`Fixed Width Text`	This style of text is used to present bindings and computer listings.
Table Text	This smaller style of text is used in tables as body text.

Functions

The function references in this guide are designed in a compatible manner for ease of reading. Each function is illustrated as follows (headings not applicable for a particular function will be omitted):

objc_draw() ⇦Function Name

WORD objc_draw(*tree, obj, depth, bx, by, bw, bh*) ⇦**Definition**
OBJECT **tree*; ⇦**Data Types**
WORD *obj, depth, bx, by, bw, bh*;

Immediately following the definition, a brief summary of the function will follow.

OPCODE The opcode related to the function will be listed in decimal and hexadecimal where appropriate.

AVAILABILITY This section will indicate any special conditions that must exist for this function to be present (i.e.: OS version, presence of **GDOS**, etc.).

PARAMETERS The meaning of each parameter to the function will be explained here. If any data pointed to by parameters is modified it will be noted here as well.

BINDING	This section will list a binding for the function in either ‘C’ format or assembly format, whichever is more appropriate. Please note bindings were written with ease of reading, not necessarily optimized code, in mind.
RETURN VALUE	This section explains the return value of the function. This covers only that value returned on the left side of the function expression.
VERSION NOTES	Under this heading, any features of a function which are only present under certain conditions are discussed.
CAVEATS	Known bugs or abnormalities of a function are listed next to this heading.
COMMENTS	Other useful information or hints are listed here.
SEE ALSO	Functions which bear a relation to the current function or which are codependent on one another are listed here.

Data Types

Within function definitions, several data types are referenced that vary from compiler to compiler. The following provides a key to the data type used and their actual definition. Other data types will contain a structure definition or ‘typedef’ within the binding. Be aware that some compilers default to 16-bit integers while others use 32-bit integers.

Usage	Synonyms	Meaning
WORD	short, int, short int	16-bit signed integer
UWORD	unsigned int, unsigned short, unsigned short int	16-bit unsigned integer
LONG	long, int, long int	32-bit signed integer
ULONG	unsigned long, unsigned int, unsigned long int	32-bit unsigned integer
VOID	void	This naming is used to denote a function with no parameters or return value.
BOOLEAN	bool, boolean, short, short int, int	16-bit signed integer valid only as **TRUE** (non-zero) or **FALSE** (0)
WORD *	short *, int *, short int *	This is a pointer to a 16-bit signed integer.
UWORD *	unsigned short *, unsigned int *, unsigned short int *	This is a pointer to a 16-bit unsigned integer.
LONG *	long *, int *, long int *	This is a pointer to a 32-bit signed integer.
ULONG *	unsigned long *, unsigned int *, unsigned long int *	This is a pointer to a 32-bit unsigned integer.
VOIDP	void *, char *	This represents a pointer to an

		undefined memory type.
VOIDPP	void **, char **	This represents a pointer to a pointer of an undefined memory type.
char *	None	8-bit character string buffer
BYTE, **CHAR**	signed byte, signed char	8-bit signed byte
UBYTE, **UCHAR**	unsigned byte, unsigned char	8-bit unsigned byte
fix31	None	This type holds a 31-bit mantissa and sign bit. The value represents the number contained multiplied times 1/65536. For a complete explanation see *Chapter 7: **VDI***.

Numeric Values

Because different computer languages use different nomenclature to specify numbers in different bases, you will come across numbers presented in a variety of different ways within this book as follows:

Prefix	Decimal 23 as an Example	Meaning
None	22	This number is shown in decimal (base 10) format. The majority of numbers shown will be in this format for simplicity.
0x	0x16	This number is shown in hexadecimal (base 16) format. Function opcodes in assembly language and numbers used as mask values will appear mostly in this format.
$	$16	Same as above.
0	026	This number is shown in octal (base 8) format. Only in extremely specialized cases will numbers by represented in this manner.
%	%00010110	This number is shownn in binary (base 2) format. Only when dealing with hardware registers and in a few other circumstances will numbers be represented in this manner.

Constant Definitions

Modern programming practices dictate the use of named constants wherever possible in place of 'raw' values. Take for example the following call to **Devconnect()**:

In 'C':

```
Devconnect( 3, 9, 0, 0, 1 );
```

In assembly language:

```
move.w      #1,-(sp)
move.w      #0,-(sp)
move.w      #0,-(sp)
move.w      #9,-(sp)
move.w      #3,-(sp)
move.w      #$8B,-(sp)
```

```
        trap        #14
        lea         12(sp),sp
```

Calling the function in this format makes debugging and program maintenance more difficult because the parameters' meanings are concealed by the numeric assignments. The following code illustrates the preferred method of coding:

In 'C':

```
/* Extracted from TOSDEFS.H, included by TOS.H */
#define ADC         3
#define DMAREC      0x01
#define DAC         0x08
#define CLK_25M     0
#define CLK_COMPAT  0
#define NO_SHAKE        1

/* Program segment */
#include <TOS.H>

Devconnect( ADC, DMAREC|DAC, CLK_25M, CLK_COMPAT, NO_SHAKE );
```

In assembly language:

```
; Extracted from TOSDEFS.I

ADC         EQU     3
DMAREC      EQU     $01
DAC         EQU     $08
CLK_25M     EQU     0
CLK_COMPAT  EQU     0
NO_SHAKE    EQU     1

Devconnect  EQU     $8B

; Program Segment
            INCLUDE     "TOSDEFS.I"

            move.w      #NO_SHAKE, -(sp)
            move.w      #CLK_COMPAT,-(sp)
            move.w      #CLK_25M,-(sp)
            move.w      #DMAREC!DAC,-(sp)
            move.w      #ADC,-(sp)
            move.w      #Devconnect,-(sp)
            trap        #14
            lea         12(sp),sp
```

Unfortunately, because many function call parameters do not have standard definitions associated with them, programmers have had to create their own, which in turn makes their programs less portable, or use the 'raw' constants. In addition, some compilers do not use standardized definitions at all.

To help alleviate these difficulties, this revision of the ***Compendium*** contains named definitions for almost every possible function parameter. These definitions come from the 'C' header files TOS.H and TOSDEFS.H or the assembly include file TOSDEFS.I, both available on disk from

SDS. Every attempt has been made to ensure that these files compile with development tools in the Lattice ‘C’, Pure ‘C’, and Alcyon ‘C’ packages. Some modifications to these files may be necessary, however, due to the peculiarities of some compilers.

The ‘C’ header files consist of two parts to improve portability between compiles. The TOS.H file is a compiler dependent file used to bind the operating system calls to definitions. This file, in turn, includes the file TOSDEFS.H which should remain portable between compilers.

When choosing definitions for inclusion in the TOSDEFS files, names given by Atari were given highest precedence followed by those assigned (and kept consistent) by compiler manufacturers. Other definitions were created with simplicity and consistency in mind.

Use of the given constants will increase program code readability and provide for a higher level of portability between compilers.