Marinetti

Version 3.0

Technical Update

"For the Apple IIGS®, the world just got a whole lot closer!"

Designed and written by Richard Bennett-Forrest

This revision: 9th June 2008

Contents

Acknowledgements 3

Introduction 4

Event triggers 5

User event triggers 6

trgUDPIn 6

System event triggers 6

strgNetworkDown 6

TCPIPGetUserEventTrigger 7

TCPIPSetUserEventTrigger 8

TCPIPGetSysEventTrigger 9

TCPIPSetSysEventTrigger 10

New DNR features 11

TCPIPGetDNRTimeouts 11

TCPIPSetDNRTimeouts 13

Link layer modules 14

LinkConfigFileName 15

Debugging and testing 16

Nifty List updates 16

New constants and equates 17

Tool error codes 17

User event trigger codes 17

System event trigger codes 17

New features and bug fixes 18

New features and enhancements 18

Bug fixes 18

Acknowledgements

Sections of this document may be based on or lifted from discussions with programmers and developers who assisted in testing Marinetti during its initial and on-going development cycles, and as such, some of their copyrighted material may have accidentally been included in this document. Any use of individually copyrighted text was unintentional and purely in the spirit of making Marinetti a reality. Concerned copyright owners should contact the author to immediately resolve any conflicts.

Development phase testers

Tony DiazAndrew Roughan
Eric Shepherd
Kelvin Sherlock
Ryan Suenaga
Geoff Weiss
Ewen Wannop

Contributors to Marinetti Open Source Project

Richard Bennett-Forrest
Tony Diaz
Glenn Jones
Andrew Roughan
Kelvin Sherlock
Michael Stephens
Ryan Suenaga
Ewen Wannop
Geoff Weiss

Introduction

Marinetti is a TCP/IP protocol suite for the Apple IIGS. It allows applications on an Apple IIGS with System 6.0.1 to connect to and interact with an internet.

The Marinetti software is free of charge, and is available from various locations, including the Marinetti Home Page:

`http://www.apple2.org/marinetti/`

Updates to Marinetti and this document are products of the Marinetti Open Source Project.

`http://sourceforge.net/projects/marinetti/`

This document describes the changes between version 2.0.1 and 3.0 of Marinetti, which are directly related to developers.

The Marinetti 2.0.1 Programmers' Guide is a prerequisite to using this document. Using both of these documents in conjunction with the Apple IIGS Toolbox Reference Manuals and widely available protocol RFCs, you should be able to add TCP/IP support to your Apple IIGS application.

Revisions since the last release of this document are highlighted with revision bars on the right hand side of the page, like this.

This documentation refers to and assumes a prior knowledge of the Apple IIGS toolbox. Apple IIGS toolbox reference manuals are available from:

Syndicomm Online Store
`http://store.syndicomm.com`

Event triggers

Marinetti version 3.0 introduces event triggers, which allow programs to receive synchronous signalling from the TCP/IP stack for specific system and `ipid` events.

Event triggers may be installed for system events, or for each individual `ipid`. When the trigger “fires”, it calls a routine in your program to indicate that the event occurred, passing several parameters on the stack. The routine is called with the accumulator and index registers long, the direct page register pointing to the Marinetti direct page, and the data bank register undefined. Upon return, the direct page and data bank registers must be preserved.

Before the call, the stack will look like this:

Previous contents	
Space	**Word** — Space for result
triggerNumber	**Word** — The number of the trigger which fired
ipid	**Word** — `ipid` or $0000 for system
— *triggerParm* —	**Long** — Parameter depending upon the trigger
	<— SP

Before returning, the stack should look like this:

Previous contents	
actionCode	**Word** — Parameter depending upon the trigger
	<— SP

`triggerNumber` The number of the trigger which fired.

`ipid` The `ipid` of the trigger which fired. If this is a system event trigger, then this value is $0000.

`triggerParm` A longword dependant upon the trigger which fired. *See the trigger descriptions below for more details.*

`actionCode` An optional parameter word returned to Marinetti indicating what it should do upon return from the trigger routine. In most cases, this parameter is unused and should be left on the stack unchanged.

Marinetti is not fully re–entrant, hence trigger routines must not call any additional Marinetti routines. In most cases, the trigger routine shall simply set a flag and return, with the main application logic detecting the flag being set and taking appropriate action within the main event loop.

Additionally, most triggers will fire within critical code within Marinetti, and the less instruction cycles used in your routine, the better.

Triggers are automatically removed from the system when the `ipid` is logged out.

User event triggers

Trigger: $0001
Trigger name: `trgUDPIn`
triggerParm: Handle to the UDP datagram.
actionCode: $0000 = Take no further action. *Default.*
$0001 = Application now owns the handle.

A UDP datagram was received. If actionCode is set to $0001 upon return, then Marinetti assumes that the application now owns the handle, and will not add it to the incoming UDP queue. Applications wishing to keep the datagram must immediately set the handle's userid field to their own.

System event triggers

Multiple applications may attempt to set a system trigger at the same time. To avoid this, applications should first check to see if a trigger is already installed, using `TCPIPGetSysEventTrigger`, before installing their own.

Trigger: $0001
Trigger name: `strgNetworkDown`
Description: Marinetti has just disconnected from the network, or has just detected that the network is already down.
triggerParm: undefined.
actionCode: undefined.

TCPIPGetUserEventTrigger $6336

Returns the requested user event trigger.

Parameters

Stack before call

Previous contents	
— *Space* —	**Long** — Space for result
triggerNumber	**Word** — Trigger to return
ipid	**Word** — Connection to use
	<— **SP**

Stack after call

Previous contents	
— *triggerProcPtr* —	**Long** — Pointer to the trigger routine
	<— **SP**

Errors	`terrBADIPID`	This `ipid` has not yet been logged in
	`terrBADTRIGGERNUM`	Invalid trigger number

BASIC

```
FUNCTION TCPIPGetUserEventTrigger (%, %) as
                    triggerProcPtr
```

C

```
extern pascal triggerProcPtr TCPIPGetUserEventTrigger
                    (Word, Word);
```

Pascal

```
function TCPIPGetUserEventTrigger (triggerNumber, ipid:
                    integer): triggerProcPtr;
```

`triggerProcPtr` This is the address of the routine that will be called by Marinetti when the trigger is fired.

TCPIPSetUserEventTrigger **$6436**

Sets the requested user event trigger. To remove the trigger, set it to zero, or logout the `ipid`.

Parameters

Stack before call

Previous contents	
triggerNumber	**Word** — Trigger to get
ipid	**Word** — Connection to use
— *triggerProcPtr* —	**Long** — Pointer to the trigger routine
	<— **SP**

Stack after call

Previous contents	
	<— **SP**

Errors	`terrBADIPID`	This `ipid` has not yet been logged in
	`terrBADTRIGGERNUM`	Invalid trigger number

BASIC

```
SUB TCPIPSetUserEventTrigger (%, %, triggerProcPtr)
```

C

```
extern pascal void TCPIPSetUserEventTrigger (Word,
                   Word, triggerProcPtr);
```

Pascal

```
procedure TCPIPSetUserEventTrigger (triggerNumber,
                   ipid: integer; tPtr:
                   triggerProcPtr);
```

`triggerProcPtr` Address of the routine to be called by Marinetti when the trigger is fired.

The routine is called in full native, with 16 bit accumulator and index registers. The direct page register points to the Marinetti direct page, and the data bank register is undefined. Upon return, the direct page and data bank registers must be preserved.

TCPIPGetSysEventTrigger $6536

Returns the requested system event trigger.

Parameters

Stack before call

Previous contents	
— *Space* —	**Long** — Space for result
triggerNumber	**Word** — Trigger to return
	<— **SP**

Stack after call

Previous contents	
— *triggerProcPtr* —	**Long** — Pointer to the trigger routine
	<— **SP**

Errors `terrBADTRIGGERNUM` Invalid trigger number

BASIC

```
FUNCTION TCPIPGetSysEventTrigger (%) as triggerProcPtr
```

C

```
extern pascal triggerProcPtr TCPIPGetSysEventTrigger
                    (Word);
```

Pascal

```
function TCPIPGetSysEventTrigger (triggerNumber:
                    integer): triggerProcPtr;
```

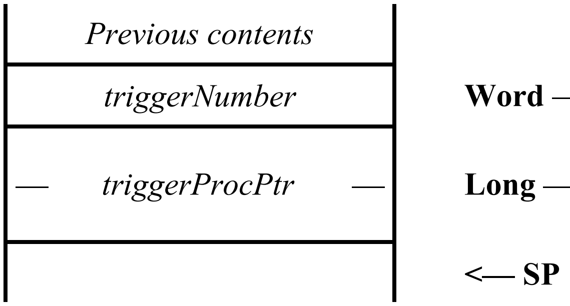
`triggerProcPtr` This is the address of the routine that will be called by Marinetti when the trigger is fired.

TCPIPSetSysEventTrigger $6636

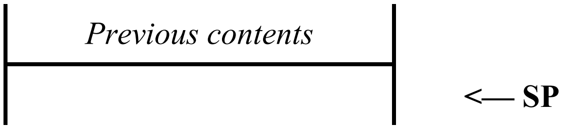
Sets the requested system event trigger. To remove the trigger, set it to zero, or logout the `ipid`.

Parameters

Stack before call



Stack after call



Errors `terrBADTRIGGERNUM` Invalid trigger number

BASIC

```
SUB TCPIPSetSysEventTrigger (%, triggerProcPtr)
```

C

```
extern pascal void TCPIPSetSysEventTrigger (Word,
                   triggerProcPtr);
```

Pascal

```
procedure TCPIPSetSysEventTrigger (triggerNumber:
                   integer; tPtr: triggerProcPtr);
```

`triggerProcPtr` Address of the routine to be called by Marinetti when the trigger is fired.

The routine is called in full native, with 16 bit accumulator and index registers. The direct page register points to the Marinetti direct page, and the data bank register is undefined. Upon return, the direct page and data bank registers must be preserved.

New DNR features

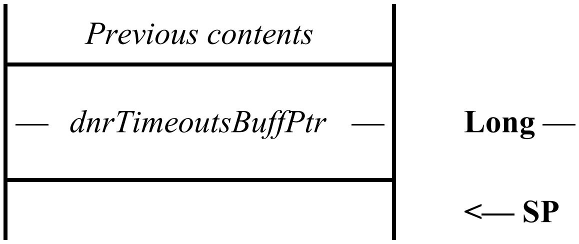
Marinetti 3.0 introduces tuning of DNR performance, via the `TCPIPGetDNRTimeouts` and `TCPIPSetDNRTimeouts` calls.

TCPIPGetDNRTimeouts $6736

Returns the current DNR timeout values.

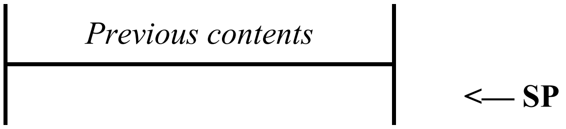
Parameters

Stack before call



Long — Pointer to 4 byte buffer for response

Stack after call



Errors None.

BASIC

```
SUB TCPIPGetDNRTimeouts (dnrTimeoutsBuffPtr)

type dnrTimeoutsRecord
    dnrRETRIES as integer
    dnrTIMER as integer
end type
type dnrTimeoutsBuffPtr as pointer to dnrTimeoutsRecord
```

C

```
extern pascal void TCPIPGetDNRTimeouts
                    (dnrTimeoutsBuffPtr);

typedef struct {
    Word dnrRETRIES;
    Word dnrTIMER;
} dnrTimeoutsStruct, *dnrTimeoutsBuffPtr;
```

Pascal

```
procedure TCPIPGetDNRTimeouts (dtPtr:
                    dnrTimeoutsBuffPtr);
```

```
dnrTimeoutsRecord = record
   dnrRETRIES: integer;
   dnrTIMER: integer;
end;
dnrTimeoutsBuffPtr = ^dnrtimeoutsRecord;
```

`dnrTimeoutsBuffPtr` Points to a 4 byte buffer for the response.

The currently defined offsets are:

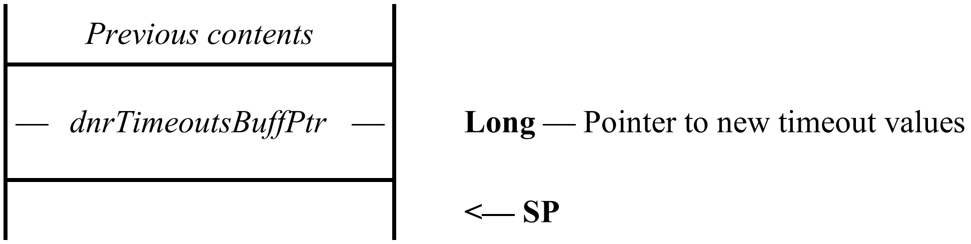
`dnrRETRIES`	+0000	How many times the DNR module retries both domain name servers. The default is 5.
`dnrTIMER`	+0002	The number of ticks before a retry times out. The default is 120 ticks (2 seconds).

TCPIPSetDNRTimeouts $6836

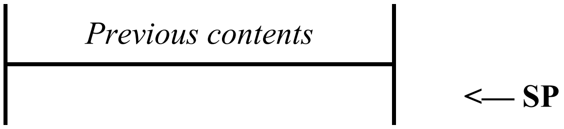
Sets new DNR timeout values.

Parameters

Stack before call



Stack after call



Errors `None.`

BASIC

```
SUB TCPIPSetDNRTimeouts (dnrTimeoutsBuffPtr)
```

C

```
extern pascal void TCPIPSetDNRTimeouts
                   (dnrTimeoutsBuffPtr);
```

Pascal

```
procedure TCPIPSetDNRTimeouts (dtPtr:
                   dnrTimeoutsBuffPtr);
```

`dnrTimeoutsBuffPtr` Points to new DNR timeout values, which Marinetti will copy into its internal DNR configuration.

See `TCPIPGetDNRTimeouts` for the definition of the buffer.

Link layer modules

Marinetti 3.0 supports several new link layer module calls, collectively known as link layer module version 2. To indicate support for these new calls, the link layer module should return the value $0002 from the `LinkInterfaceV` call.

Marinetti 3.0 ships with version 2 interface versions of all its link layer modules, and supports third party modules at both version 1 and 2.

LinkConfigFileName $0018

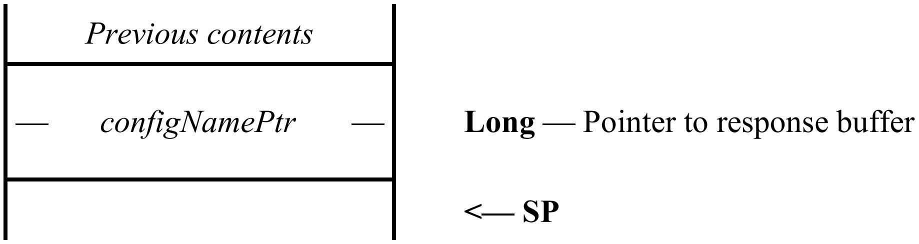
The configuration data for link layer modules is stored in the resource fork of a single configuration file inside the TCPIP folder. While Marinetti starts and shuts down the Resource Manager as required, this could cause excessive CPU overhead as well as data corruption due to bugs in the Resource Manager.

In future Marinetti will store configuration data in a file for each link layer module.

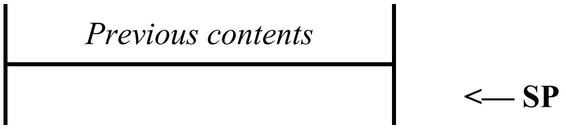
This call returns the filename of the configuration file as defined by the link layer module.

Parameters

Stack before call



Stack after call



`configNamePtr` Pointer to a 16 byte buffer into which the ProDOS filename will be returned as a pString.

Debugging and testing

Nifty List updates

Marinetti 3.0 contains additional tool calls and error codes, which should be applied to the `NList.Data` file, following the definitions for Marinetti 2.0.1 that are documented in the Marinetti 2.0.1 Programmers Guide.

The new tool call definitions are as follows:

```
6336 TCPIPGetUserEventTrigger(triggerNumber,ipid):triggerProcPtr/4
6436 TCPIPSetUserEventTrigger(triggerNumber,ipid,@triggerProcPtr)
6536 TCPIPGetSysEventTrigger(triggerNumber):triggerProcPtr/4
6636 TCPIPSetSysEventTrigger(triggerNumber,@triggerProcPtr)
6736 TCPIPGetDNRTimeouts(@dnrTimeoutsBuffPtr)
6836 TCPIPSetDNRTimeouts(@dnrTimeoutsBuffPtr)
```

The extra error codes are:

```
3621 terrBADTRIGGERNUM
```

Once the changes have been made, save them back to disk and reboot. You should now be able to issue Nifty List commands against the Marinetti tool calls and error codes. If issuing calls outside of your application, you will most likely need to use Nifty List to issue the `_LoadOneTool(36,300)` call first.

A file containing the above Nifty List configuration for Marinetti can be found in the Marinetti Open Source Project CVS Repository:

> `http://marinetti.cvs.sourceforge.net/marinetti/MOSP/Tools/NiftyList/nl.marinetti`

GSBug

GSBug has the ability to view data structures in memory by using templates. A template file for Marinetti can be found in the Marinetti Open Source Project CVS Repository:

> `http://marinetti.cvs.sourceforge.net/marinetti/MOSP/Tools/GSBug/tcpip.template`

For ease of access, you may wish to copy the template file to your `*:System:System.Setup` directory.

Refer to the GSBug documentation for more information on how to use this file.

New constants and equates

New tool error codes

`terrBADTRIGGERNUM`	`$3621`	Invalid trigger number

User event trigger codes

`trgUDPIn`	`$0001`	UDP datagram received

System event trigger codes

`strgNetworkDown`	`$0001`	The network has gone down

New Features and bug fixes

New features and enhancements

Marinetti 3.0 includes the following new features and enhancements.

- Link layer module configuration data is now stored in the `*:System:TCPIP:` folder with a name specified by the link layer module.
- New link layer version (2) to support new configuration data files.
- PPP link layers now support the CHAP Authentication Protocol.
- PPP link layers now support the MS-CHAPv1 Authentication Protocol.
- MacIP link layer is now included.
- Uthernet link layer is now included.

Bug fixes

Marinetti 3.0 includes fixes for the following bugs in Marinetti 2.0.1.

- There was a stack imbalance sending ICMP packets. Intermittent results.
- There was a stack imbalance when querying the status of UDP packets.
- There was a stack imbalance in incoming SYN error handler. TCPLogin would fail intermittently.
- There was a memory overrun in the TCPRead call when specifying a buffer length to be used and the available data exceeded this length when the socket was closed.
- There was a logic problem with processing TCP packets that arrived out of sequence. Marinetti ignored any incoming data if the socket was closed by the host before all packets had been processed.
- There was incorrect handling of escaped characters in the PPP link layers. Packets containing escaped characters failed the CRC check and the packet would be dropped. This would result in hangs, and loss of data when using the PPP link layers.
- LCPTerminateRequest was not supported by the PPP link layers. This would result in SysFail errors during negotiation of connections with some hosts.
- There was a pointer arithmetic bug in the TCP TimerData routine. This would result in SysFail errors within the PtrToPtr routine.