F00D Processor

From Vita Development Wiki

This processor is hypothesized to perform most of the cryptography tasks including storing and handing of keys. There is little information about it though. The **F00D Processor** (named after the `e_machine` field of the ELF headers) is likely a custom Toshiba MeP (http://en.wikipedia.org/wiki/Media-embedded_processor) core.

You can find an open source f00d protocol implementation at https://github.com/xyzz/f00d

Contents

Communication

Communication seems to go through some sort of FIFO register.

Write

To write, put the double word into `0xE0000010`. Next read `0xE0000010` until it returns 0, which indicates the data was read by the F00D processor.

Read

To read, get a double word from `0xE0000000`. If it returns 0, no data is available. Otherwise, acknowledge that the data has been read by putting the same data into `0xE0000000`.

Extra ports

In addition to the 0xE0000000 and 0xE0000010, the communication with F00D seems to happen with other ports too.

Port	Read	Write	Used by
0xE0000004	YES	?	SMC 0x12d, 0x136, 0x137, 0x13B Interrupt 0xC8
0xE0000008	YES	?	SMC 0x12d, 0x136, 0x137, 0x13B Interrupt 0xC8
0xE000000C	YES	?	SMC 0x12d, 0x136, 0x137, 0x13B Interrupt 0xC8
0xE0000014	YES	YES	Used to send commands to f00d secure module, see Executing sm commands
0xE0000018	YES	YES	Used to send commands to f00d secure module, see Executing sm commands
0xE000001C	YES	YES	Used to send commands to f00d secure module, see Executing sm commands
0xE0000054	?	YES	SMC 0x12d, 0x135, 0x13B, Interrupt 0xC8
0xE0000058	?	YES	SMC 0x12d, 0x135, 0x13B, Interrupt 0xC8
0xE000005C	?	YES	SMC 0x12d, 0x135, 0x13B, Interrupt 0xC8
0xE0010000	YES	YES	Used by F00D Command 0 to prepare F00D to receive an address that contains the secure kernel enp.
0xE0010004	YES	?	Used by F00D Command 0 to check if the value written in `0xE0010000` is okay (it should return a value <= 0)

Protocol

A 32-bit command buffer is defined below. The command is sent to the F00D processor with the method listed above.

Bit End	Bit Start	Name	Description
31	23	?	?
22	19	flag?	b1010, b0010, b0001
18	13	?	Always 0
12	8	ID	Command ID
7	1	?	Always 0
0	0	Valid	Set 1 to indicate command is valid

Command ID

There are a total of 14 commands. Below are notes on different commands. Commands pass data through the 0x100 shared buffer (set up in command 0).

0x0

This command is used to setup a shared buffer with F00D. First F00D will write in `0xE0000000` either 0x200 or 0x600 (and if an error happened, 0x400 ?). If the value 0x600 is written, it will first setup a buffer that contains the f00d secure kernel enp. First it will write 0x1 to `0xE0010000` and after that 0x0 to `0xE0010000`, next, it will wait `0xE0010000` return 0 and `0xE0010004` return a value <= 0x0. If everything goes alright, it will clear the low 2 bits of the f00d secure kernel enp address (normally at 0x1F850000) and OR it with 0x1 and then write it to `0xE0000010`. After it, the process is the same that the 0x200. If the value 0x200 is written, it will set the 0x100 sized shared buffer. First the physical address of the buffer is written to `0xE0000010` and then command 0x0 is written.

0x1

This command is called by a function inside the SceSblSmSchedForTZS_0xE72F2886 or SceSblSmsched_start (if an error happened while SceSblSmsched is initializing). This command seem to be used to ask the f00d to remove all resource allocated (like shared memory address, secure module loaded, etc), and probably power off or reset it.

0x2

Load sm.

Offset	Size	Description
0x0	0x4	num_paddrs
0x4	0x4	paddr_list for sm elf
0x8	0x4	buf_0x40: some 0x40 buffer
0xC	0x4	ctx_0x4
0x10	0x4	ctx_0x8
0x14	0x4	ctx_0xC
0x1C	0x4	field_60: system version (2)
0x20	0x4	partition ID

0x3

Load previously suspended sm

Offset	Size	Description
0x0	0x4	num_paddrs
0x4	0x4	paddr_list for suspend buffer
0x8	0x4	buf_0x40: some 0x40 buffer
0xC	0x4	delayed_cmd

0x4

Suspend current sm

Offset	Size	Description
0x0	0x4	num_paddrs
0x4	0x4	paddr_list for suspend buffer
0x8	0x4	buf_0x40: some 0x40 buffer
0xC	0x4	delayed_cmd

0x5

Kill current sm

0x6

Force stop? Called from `sceSblSmSchedDeleteAll`.

0x7

Unused.

0x8

Unused.

0x9

Set a 0x80 sized shared buffer.

Offset	Size	Description
0x0	0x4	paddr
0x4	0x4	length

0xA

Set the SCE encrypted revocation list.

Offset	Size	Description
0x0	0x4	num_paddrs
0x4	0x4	paddr_list for revocation list

0xB

0xC

0xD

0xE

0xF

0x10

Interrupts

There are 4 interrupts that f00d sends. There are only two interrupt handles, however.

- `cry2arm0`: handles interrupt 200. This is notifications about current sm status changes (loaded, unloaded, suspended, etc)
- `cry2arm123`: handles interrupts 201, 202, 203. This interrupt is sent once an sm command has finished processing (see below).

Executing sm commands

When sm is loaded, you can execute a command it provides by writing into registers 0xE0000014, 0xE0000018, 0xE000001C paddr to the command buffer OR'd with 1.

Command buffer structure:

Offset	Size	Description
0x0	0x4	Size of header + data
0x4	0x4	Command ID
0x8	0x4	unk1
0xC	0x4	unk2
0x10	0x30	padding
0x40	variable, max=0x1000-0x40	data buffer

Once result is available, ARM will get an interrupt 201-203. Read status code from 0xE0000004 (or 0xE0000008, 0xE000000C) and confirm you received it by writing it back to the same register.

Possible results:

- 1: command executed successfully
- 3: invalid command ID
- 5: invalid command buffer paddr (e.g. trying to pass secure memory)
- 9: SCE_SBL_ERROR_COMMON_EIO

Memory

`kprx_auth_sm.self` is allowed access to `0x1F000000`, `0x1F840000`, `0x20000000`, and `0x40300000`. The address checks is likely done in software. F00D has it's own private 128KB memory from `0x00800000` to `0x00820000`. F00D SELFs are typically loaded to `0x0080B000`.

Task states

ARM secure world implements a scheduler for SM tasks. Normally, only one task can run on f00d at a time. The external scheduler running on ARM allows to have multiple tasks at once. The following provides an explanation of various states the task can be in.

ID	Description
1	Created / Suspended (Unloaded)
2	Running
3	Errored (Unloaded)
4	? (Unloaded)
5	? (Unloaded)
6	Start/resume requested
7	Suspending
8	?
9	?
10	?
11	Suspend requested
12	?