Contents

Kernel UART commands....1
Read raw....1
Write Raw....2
Create Object....2
Delete Object....3
Read Object....3
Write Object....3
Create Process....3
Run Process....3
Stop Process....4
Get Length....4
Kill Process....4
Read File....4
Write File....4
Delete File....5
Get Catalog....5
Erase All....5
Get Status....5
Write Blocked....6
CPU List....6
Get Performance....6
Read Breakpoint....6
Process List....6
CPU prefix....7
Binary transfer prefix....7

Kernel UART commands

Commands are transmitted through the debug UART in text form, which makes it possible, if necessary, to assess the state of the system using a simple terminal in manual mode. CoreExplorer software uses these commands to communicate with the processor system to start and stop processes, transfer data, and debug software.

Read raw

RR <address> <count>

The command specifies the physical start address and the number of bytes. If the number of bytes is omitted, then a block of 256 bytes will be transmitted. Data are output in HEX, 16 bytes per line. Lines are separated by bytes 0Dh and 0Ah, bytes inside lines - by spaces.

Example:

Input:

RR 1000 67

Kernel output:
RR
22 10 08 00 43 36 0F 80 35 83 00 85 43 48 23 25
43 36 7F 80 33 38 25 23 0F E5 2F E0 0F E0 40 EE
27 22 22 40 0F ED 3F E0 0F E0 44 E0 E0 D4 44 D0
CF 34 0F 40 35 43 E0 46 FC 42 B0 15 44 40 0F 35
0F 00 35
›

Data output always begins with a repeat of the command, followed by a line feed and data output begins. Issuance ends with line feed and prompt ">"

Write Raw

WR <address> <count>CR <data>

The command allows you to write a string of bytes starting from the desired physical address. First you need to specify the address and the number of bytes. After receiving this information, the kernel will give readiness to receive data in the form of a repeat of the WR command / but without parameters / After that, you can enter data. Bytes are entered in HEX, leading zeros in bytes (from 00 to 0F) can be skipped when entering. Bytes must be separated by spaces. Reception will complete either when all the data bytes have been received, or if a character other than a space and characters 0-9 and A-F is entered.

Create Object

CO <size> <access rights> <task ID>

Create an object with the specified length, control byte, and Task ID parameter. Control Byte and Task ID are optional parameters. The length is specified in bytes, the control byte and Task ID are specified in HEX. As a response, Kernel returns the value of the object selector in HEX. Example command:

>CO 1000

Kernel output:

CO 0000000000000021

>

The selector in this case is displayed as a 64-bit number in HEX.

Delete Object

DO <selector>

The command is used to delete any object, regardless of its parameters. The selector is specified in HEX. If the command is executed correctly, Kernel returns an acknowledgment in the form of a command retry, but without the selector. For example:

>DO 21

Kernel output:

DO

>

Read Object

RO <selector> <offset> <count>

Reading data from an object. The selector, offset and number of bytes are determined. The response is issued in the same way as in the case of the RR command. The selector and offset must be in HEX, and the counter must be in decimal.

Write Object

WO <selector> <offset> <count> <data>

Writing data to an object.

Create Process

CP <selector> <parameter>

Process creation. The command parameters are the object selector and the parameter passed to the process. Both command parameters must be specified in HEX. The start parameter is optional. It is entered into the PSO of the process and can be retrieved with a GETPAR instruction. The selector must point to an object containing a block describing the process, see ***process files.pdf*** When the command is executed, this object is expanded into a set of objects necessary for the process to function. The minimum set of objects is:

- PSO;
- One code object as minimum;
- • Four objects of software stacks for four levels of privilege.

Upon successful execution of the command, the kernel returns the CP command code to the terminal. The created process is in a stopped state, it does not receive control and its selector is not listed in the list of active processes. The following command is used to activate the process.

Run Process

RP <selector>

Launching the process for execution. The parameter of the command is the value of the selector of the process object. This is not a PSO selector, it is an object selector with the structure described in ***process files.pdf*** The PSO selector, if a process is created, is retrieved from the process description object and entered into the list of active processes.

Stop Process

SP <selector>

Stop the execution of the process. The command parameter specifies the selector of the process description object. The PSO selector is retrieved from this object, looked up in the list of active processes and removed from it, if any.

Get Length

GL <selector>

Request for the length of an object. The command returns to the terminal the length of the object in bytes, in decimal code.

Kill Process

KP <selector>

Delete the process and all objects created by it. You can delete a process only if it is inactive. Those. you must first stop the process by excluding it from the list of active processes and only then can you delete it. When deleting a process, not only objects created when the process was created are removed from system memory, but also all other objects created by the process itself during operation.

Read File

RF <name>

Reading data from a file located in SPI FLASH. The file is placed into an object. The command prints the following text messages to the terminal:

- 'File not found' - if the record with the specified name is not in FLASH;
- 'Not enough memory' - if there is no space in the RAM to create an object for the required size;
- 'CRC Error' - if a data reading error occurred, the calculated CRC and the CRC specified in the FLASH sector header did not match;
- 'File read into object:' <selector> - if the file is read correctly, the selector of the created object is indicated.

Write File

WF <selector> <name>

Writing an object to FLASH as a file. The command parameters specify the object selector in HEX and the file name. The following messages may appear in response to the command:

- ‘WF’ - if the file was written successfully;

- 'File already exists' - if a file with the specified name is already present;
- 'Not enougth free size' - there is no place in FLASH to write the object.

Delete File
DF <name>

Delete file in FLASH. Possible command responses:

- `DF' - if the file was successfully deleted;
- 'File not found' - if there is no specified file.

Get Catalog
GC

The command to read the directory of files written to FLASH. Example command response:

```
0000000000000062
SimpleLED
TwiddleGenerator
SineGenerator
FFTProcessor
FFTManager
SimpleFFT
Fonts
Services
›
```

In the first line, the kernel displays in the HEX code the total length of the data, including the characters 0Dh and 0Ah. This is followed by a list of filenames separated by 0Dh and 0Ah characters.

Erase All
EA <Flash ID >

Clear Flash completely. Flash ID is served so that there are no accidental command entries. For example, for W25Q128FV this code should be 17EF.

Get Status
GS

System status request command. Sample response:

```
›GS
Platform: EP2AGX125EF29I3
Processor core: 1x16V0
Kernel built: 30.05.2021 23:12:45
Main clock: 100MHz
Total free memory: 1073732480
Cached free memory: 1073732480
Empty descriptors: 70
Free memory descriptors: 2
Object descriptors: 22
Stream descriptors: 2
Active processes:
0000000000000008
```

Below the line “Active processes”, the PSO selectors of the processes that are in the active list are listed.

Write Blocked

WB <selector> <offset> <count> <data>

Writing data to the object is blocked. Hardware interrupts and automatic process switches are blocked during data transfer.

CPU List

CL

List the processors in the system and their numbers. Example command:

```
›CL
0000000000000005
0D
›
```

First, the length of the data is given in bytes, in HEX. In addition to the symbols of processor numbers, the length also includes symbols 0Dh, 0Ah + one symbol ‘>’. In the given example, there is only one core in the system and it has a network number of 0Dh.

Get Performance

GP

Reading performance monitor readings. Sample response:

```
›GP
0000000000000B0F 000013930000000B
›
```

2 64-bit words in HEX are output. The first word contains only the current performance value, the second word contains the minimum number of commands in the lower 32-bits and the maximum number of commands in the upper 32-bits for the specified measurement period.

Read Breakpoint

RB

Reading breakpoints. First, the kernel issues a 64-bit word in HEX defining the number of entries in the checkpoint table, and then, if there are entries, 64-bit words in HEX are displayed, which indicate the PSO selector of the process in which the BKPT command was triggered (least significant 32 bits) and the offset of the BKPT command itself (high 32 bits)

Process List

PL

The command allows you to get a list of processes registered in the system. Example command response:

```
>PL
000000000000002D Services 0000000000000006 000000000000000D
0000000000000031 FourStageLED 000000000000001A 0000000000000000
0000000000000000
›
```

Each line starts with a 64-bit value of its length. If there is no more data, then the first word in the string will be 0. If the length is not zero, then the process name follows, followed by the value of the selector of the process object and the value of the selector PSO. If the PSO value is 0, it means that this process has not been created. For example, after giving the command

```
›CP 1A 6
CP
›PL
000000000000002D Services 0000000000000006 000000000000000D
0000000000000031 FourStageLED 000000000000001A 000000000000001B
0000000000000000
›
```

The FourStageLED process will already have a valid PSO selector equal to 1Bh.

CPU prefix

The command can be addressed to another core of a multi-core system. Such commands and their responses pass through the core to which the debug UART is connected. In order for the command to be transmitted in transit to another core of the multiprocessor network, it is necessary to determine the processor prefix.

@CC LLLL[command body or data]

Where CC is the processor number in HEX, LLLL is the number of bytes in HEX that makes up the command being transmitted.

Binary transfer prefix

RR, WR, RO, WO, WB commands can transfer data not only in text format, but also in binary form. The CPU Observer program uses a binary format for data transfer, i.e. data bytes are not recoded to text format and back during transmission via the debug UART. This speeds up data transfer. Binary data transfer prefix - the 'R' character is placed before the first command parameter, for example:

RR R00100 45

This command requests reading 45 bytes starting from address 100h and transmitting them in binary form.