Author: MCUXpresso IDE Team

Version: 1.0.0 24th Feb 2020

Debugging Kinetis KE0x with MCUXpresso IDE v11.1.1 (LinkServer)

Table of Contents

1 Background

LinkServer / CMSIS-DAP Debug support for the large range of Kinetis MCUs relies on the use of a common set of features. This allows the management of watchdogs, flash programming interfaces, memory footprints etc. to be handled by a small set of flash drivers and a single debug connection script. While there are a number variations within these MCUs, these have been accommodated by a single LinkServer Connect script (kinetisconnect.scp) and small range of semi generic Flash Drivers.

Tip: The majority of Kinetis flash drivers, for example the FTFE_4K.cfx, are 'smart drivers' that perform some interrogation of the MCU under debug and so can support a range of Kinetis MCUs.

Kinetis flash drivers are typically named based on the Flash Memory Module and the flash sector size (which unfortunately cannot be read from internal registers within these parts). Where the supported Flash Memory Modules previously only included: FTFE, FTFA, FTFL.

However, the range of KE0x MCUs use previously unsupported Flash Memory Module types, known as FTMRE and FTFRH. These smaller footprint parts also provide very limited ability for self configuration. In addition, these devices require different Watchdog management and debug configuration.

The range of SDKs for these devices, such as SDK_2.x_FRDM-KE04Z (v2.7.0 and earlier) erroneously reference LinkServer connection scripts and flash drivers that will not work with these MCUs.

This document explains how KE0x projects can be updated to reference the correct flash drivers and connect scripts provided by **MCUXpresso IDE v11.1.1 and later**. Furthermore, it shows how an SDK can be modified to address this oversight until such time as the SDKs for these MCUs are updated.

Note: This information below only applies if a LinkServer/CMSIS DAP debug solution is to be used (SEGGER J-Link and PE Micro debug connections manage these devices via their own internal configuration files).

Finally, it is assumed that the reader of this document is familiar with the operation of MCUXpresso IDE and SDKs. If not, please be sure to refer to the MCUXpresso IDE User Guide.

2 Kinetis OpenSDA CMSIS-DAP Debug Firmware

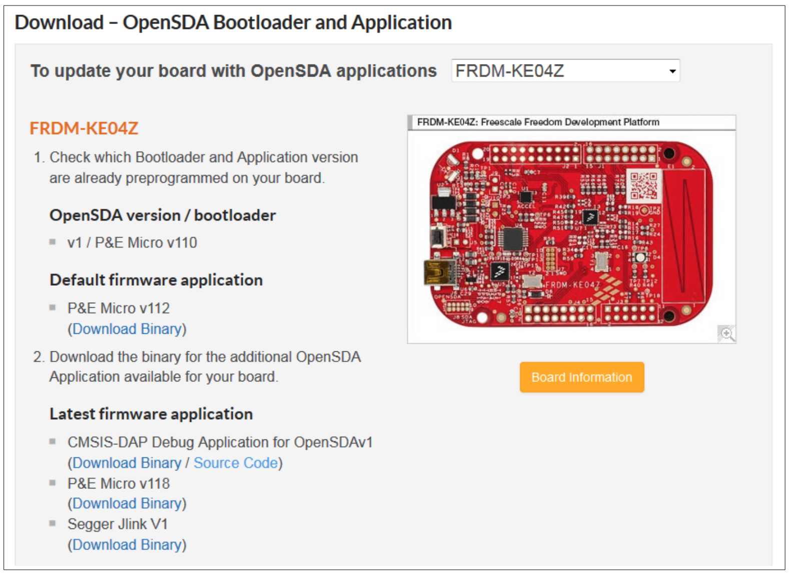
Tip: these boards have two 10 way debug connection mount points (typically without fitted headers), one for debug of the OpenSDA Debug processor and one for the target KE0x device. A debug header can be soldered to the board's SWD mount points for debug via an external debug probe such as the LPC-Link2. The LPC-Link2 can also be used to power to target processor via the LPC-Link2 jumper JP2.

Freedom development boards supporting these devices ship with OpenSDA debug circuit that offers serial support only.

LinkServer/CMSIS-DAP OpenSDA debug support can be added as follows:

From MCUXpresso IDE go to Help -> Additional resources -> OpenSDA Firmware Updates, this will open a web page 'OPENSDA: OpenSDA Serial and Debug Adapter'.

From this page, locate the dropdown and select the required FRDM-KE0x board, the FRDM-KE04Z board is shown in the example below:


Download – OpenSDA Bootloader and Application

To update your board with OpenSDA applications FRDM-KE04Z

FRDM-KE04Z

1. Check which Bootloader and Application version are already preprogrammed on your board.

OpenSDA version / bootloader
- v1 / P&E Micro v110

Default firmware application
- P&E Micro v112 (Download Binary)

2. Download the binary for the additional OpenSDA Application available for your board.

Latest firmware application
- CMSIS-DAP Debug Application for OpenSDAv1 (Download Binary / Source Code)
- P&E Micro v118 (Download Binary)
- Segger Jlink V1 (Download Binary)

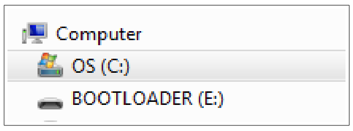
FRDM-KE04Z: Freescale Freedom Development Platform

Board Information

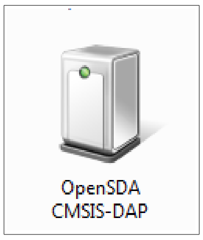
Click to download the latest CMSIS-DAP OpenSDA binary file and extract the binary file (s19S from the zip.

To install the firmware, follow the procedure below:

1. Power off the board
2. Press and hold the Reset Switch
3. Connect a USB cable to the OpenSDA connector
4. Release the Reset Switch
5. Open a filer window and observe a drive labelled BOOTLOADER appears:



6. Open this drive and drag the previously downloaded firmware file onto the filer window
7. Once complete, eject the device
8. Power off the board
9. Re-power the board and observe the board enumerates as below:



Tip: Under Windows 10, OpenSDA Bootloaders might experience problems and the OpenSDA LED will blink an error code. The following article discusses the problem and how it can be fixed: https://mcuoneclipse.com/2018/04/10/recovering-opensda- boards-with-windows-10

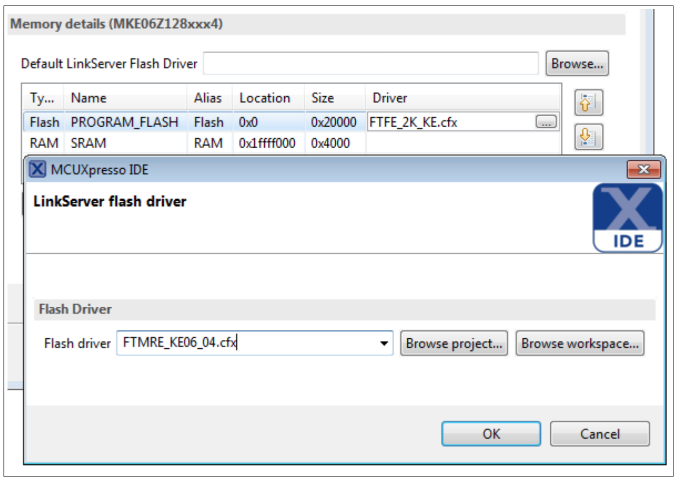
3 Flash Drivers

From MCUXpresso IDE version 11.1.1, the following LinkServer/CMSIS-DAP flash drivers for KE0x MCUs are shipped with the product.

Part	Flash KB	Sector B	RAM KB	Base	Type	Driver	Board
KE06	128	512	16	0x1FFFF000	FTMRE	FTMRE_KE06_04.cfx	FRDM Kinetis KE06
KE06	64	512	8	0x1FFFF800	FTMRE	FTMRE_KE06_04.cfx	NA
KE04	128	512	16	0x1FFFF000	FTMRE	FTMRE_KE06_04.cfx	NA
KE04	64	512	8	0x1FFFF800	FTMRE	FTMRE_KE06_04.cfx	NA
KE04	8	512	1	0x1FFFFF00	FTMRE	FTMRE_KE04_Tiny.cfx	FRDM Kinetis KE04
KE02	64	512	4	0x1FFFFC00	FTMRH	FTMRH_KE02.cfx	FRDM Kinetis KE02
KE02	64	512	2	0x1FFFFE00	FTMRH	FTMRE_KE02_Tiny.cfx	NA

The range of SDKs for these devices, such as SDK_2.x_FRDM-KE06Z v2.7.0 (and earlier) erroneously reference LinkServer flash drivers that will not work with these MCUs, therefore the correct flash driver must be selected for any new project or imported example.

This can be done per project by Right Clicking on the project and selecting *Project Properties → C/C++ Build → MCU settings*. From here you can click within the Driver column and select the correct driver as listed in the Table above.



Memory details (MKE06Z128xxx4)

Default LinkServer Flash Driver

Ty...	Name	Alias	Location	Size	Driver
Flash	PROGRAM_FLASH	Flash	0x0	0x20000	FTFE_2K_KE.cfx
RAM	SRAM	RAM	0x1ffff000	0x4000	

MCUXpresso IDE

LinkServer flash driver

Flash Driver

Flash driver: FTMRE_KE06_04.cfx

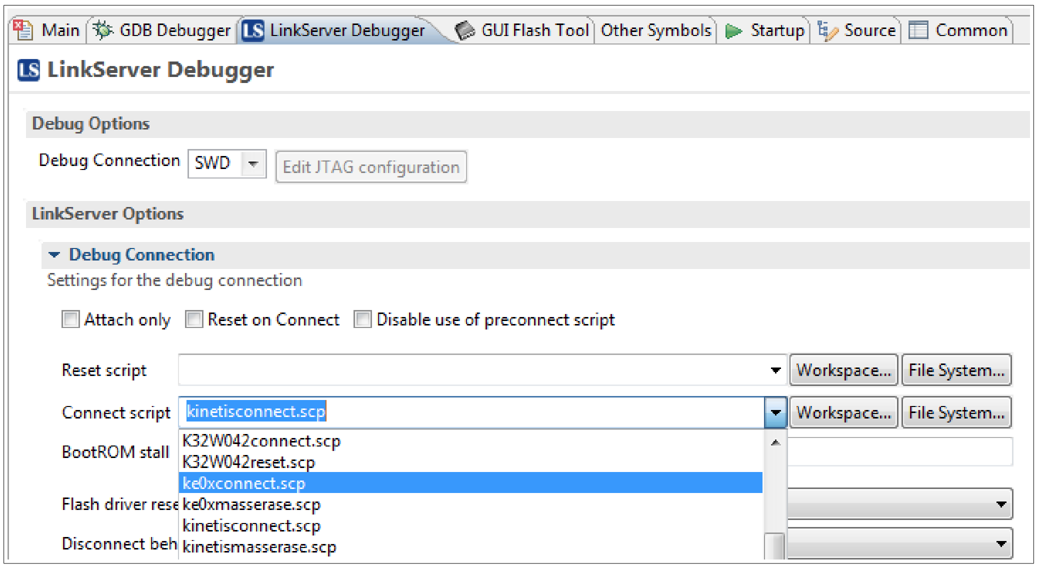
4 Connect Scripts

The range of SDKs for these devices, such as SDK_2.x_FRDM-KE06Z v2.7.0 (and earlier) also erroneously reference LinkServer Connect Scripts that will not work with these MCUs, therefore the correct Connect Script must be set.

Connect Scripts are referenced from a project's Debug Launch Configuration. Therefore a LinkServer Launch Configuration must first be manually created.

This can be done per project by Right Clicking on the project and selecting *Launch Configurations → Create new... → MCUXpresso IDE LinkServer (inc. CMSIS-DAP) probes*.

Once created, Double Click to open the Launch Configuration and manually select the correct Connect Script, as show below.



LinkServer Debugger

Debug Options

Debug Connection SWD

Edit JTAG configuration

LinkServer Options

Debug Connection

Settings for the debug connection

☐ Attach only ☐ Reset on Connect ☐ Disable use of preconnect script

Reset script

Connect script kinetisconnect.scp

BootROM stall

Flash driver rese

Disconnect beh

K32W042connect.scp
K32W042reset.scp
ke0xconnect.scp
ke0xmasserase.scp
kinetisconnect.scp
kinetismasserase.scp

5 Updating an Existing SDK to specify the correct settings

An SDK is described from an internal XML file known as the Manifest – this file can be edited to reference the appropriate flash driver and connect script. Once this change is made, new projects and imported examples will pickup this new information.

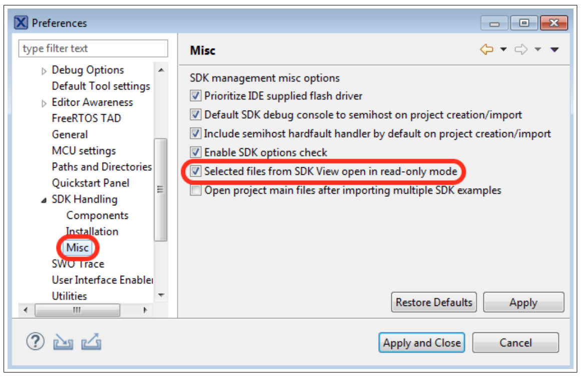
Although, SDKs can potentially be handled outside of the IDE, the description below will manage this operation entirely with the IDE itself.

First an SDK for the KE0x MCU of interest must be installed into the IDE.

Note: Since an internal file within the SDK is to be edited it must be unzipped, therefore the SDK should be obtained from the SDK Builder site rather than via the SDK Install View within the IDE. SDKs from the SDK Install View are managed as Eclipse plugins and so cannot be unzipped.

From the Installed SDKs view, right click on the SDK and select the option Unzip Archive.

Next, the IDE Preference must be unchecked to allow SDK internal files to be opened in Read Write mode as shown below:

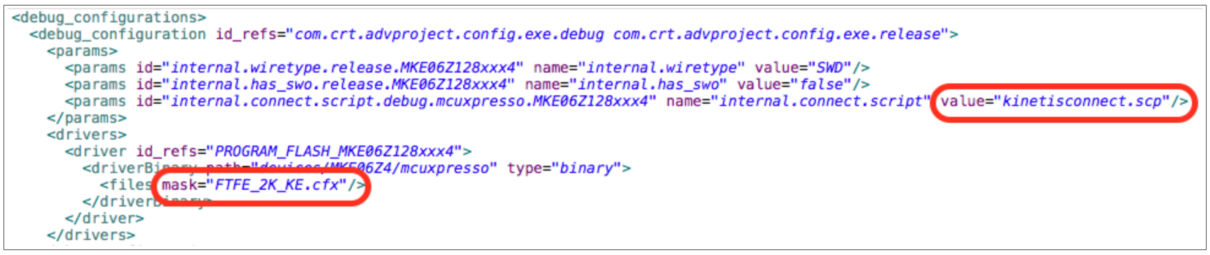


SDK management misc options

- ☑ Prioritize IDE supplied flash driver
- ☑ Default SDK debug console to semihost on project creation/import
- ☑ Include semihost hardfault handler by default on project creation/import
- ☑ Enable SDK options check
- ☑ Selected files from SDK View open in read-only mode
- ☐ Open project main files after importing multiple SDK examples

Again from the Installed SDK View, right click on the unzipped SDK and select SDK Info → Show XML Description. This action will open the SDKs internal XML manifest within the IDE's editor view.

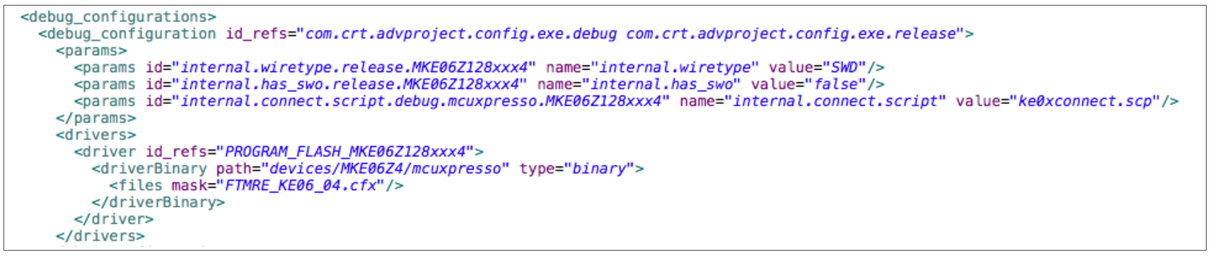
This view has two internal tabs, ensure that the Source tab is selected. Search for the existing flash driver by name (or part name) and see the two locations that require editing.



```
<debug_configurations>
  <debug_configuration id_refs="com.crt.advproject.config.exe.debug com.crt.advproject.config.exe.release">
    <params>
      <params id="internal.wiretype.release.MKE06Z128xxx4" name="internal.wiretype" value="SWD"/>
      <params id="internal.has_swo.release.MKE06Z128xxx4" name="internal.has_swo" value="false"/>
      <params id="internal.connect.script.debug.mcuxpresso.MKE06Z128xxx4" name="internal.connect.script" value="kinetisconnect.scp"/>
    </params>
    <drivers>
      <driver id_refs="PROGRAM_FLASH_MKE06Z128xxx4">
        <driverBinary path="devices/MKE06Z4/mcuxpresso" type="binary">
          <files mask="FTFE_2K_KE.cfx"/>
        </driverBinary>
      </driver>
    </drivers>
```

Edit the file mask to contain the updated flash driver name, which in this example will be called FTMRE_KE06_04.cfx.

Edit the connect script value to contain the required connect script – in this example the ke0xconnect.scp script should be used.



```
<debug_configurations>
  <debug_configuration id_refs="com.crt.advproject.config.exe.debug com.crt.advproject.config.exe.release">
    <params>
      <params id="internal.wiretype.release.MKE06Z128xxx4" name="internal.wiretype" value="SWD"/>
      <params id="internal.has_swo.release.MKE06Z128xxx4" name="internal.has_swo" value="false"/>
      <params id="internal.connect.script.debug.mcuxpresso.MKE06Z128xxx4" name="internal.connect.script" value="ke0xconnect.scp"/>
    </params>
    <drivers>
      <driver id_refs="PROGRAM_FLASH_MKE06Z128xxx4">
        <driverBinary path="devices/MKE06Z4/mcuxpresso" type="binary">
          <files mask="FTMRE_KE06_04.cfx"/>
        </driverBinary>
      </driver>
    </drivers>
```

Now save the file and click the icon in the Installed SDK view to refresh the SDK Part Support.

NOTE: These changes will only be seen in newly created Projects, newly imported examples and new launch configurations.

6 Debug Operation

A successful debug operation will show the use of the updated Connect Script and flash driver. Below shows the debug log from a Kinetis Freedom KE04 board via the LinkServer OpenSDA Debug Connection.

```
MCUXpresso IDE RedlinkMulti Driver v11.1 (Feb  4 2020 08:54:29 - crt_emu_cm_red-
link build 9)
Found part description in XML file MKE06Z4_internal.xml
Reconnected to existing LinkServer process.
============= SCRIPT: ke0xconnect.scp =============
KE0X Connect Script
Probe Handle 1 Open
DpID = 0BC11477
Assert NRESET
Reset pin state: 00
Power up Debug
APID = 0x04770031
MDM-AP APID: 0x001C0020
Release NRESET
Reset pin state: 01
MDM-AP System Reset/Hold Reset/Debug Request
MDM-AP Control: 0x0000001C
MDM-AP Status (Flash Ready) : 0x00000002
Part is not secured
MDM-AP Control: 0x00000014
MDM-AP Control (Debug Halt): 0x00000004
MDM-AP Core Halted
MDM-AP Status (Debug Halt): 0x0001000A
Disable Watchdog
============= END SCRIPT =========================
Probe Firmware: LPC-LINK2 CMSIS-DAP V5.361 (NXP Semiconductors)
Serial Number:  IQCYAWEV
VID:PID:  1FC9:0090
USB Path: \\?\hid#vid_1fc9&pid_0090&mi_00#8&149efaae&0&0000#{4d1e55b2-f16f-11cf-
88cb-001111000030}
Using memory from core 0 after searching for a good core
debug interface type      = Cortex-M0+ (DAP DP ID 0BC11477) over SWD TAP 0
processor type            = Cortex-M0+ (CPU ID 00000C60) on DAP AP 0
number of h/w breakpoints = 2
number of flash patches   = 0
number of h/w watchpoints = 2
Probe(0): Connected&Reset. DpID: 0BC11477. CpuID: 00000C60. Info: <None>
Debug protocol: SWD. RTCK: Disabled. Vector catch: Disabled.
Content of CoreSight Debug ROM(s):
RBASE F0002000: CID B105100D PID 000008E000 ROM (type 0x1)
ROM 1 E00FF000: CID B105100D PID 04000BB4C0 ROM (type 0x1)
ROM 2 E000E000: CID B105E00D PID 04000BB008 Gen SCS (type 0x0)
ROM 2 E0001000: CID B105E00D PID 04000BB00A Gen DWT (type 0x0)
ROM 2 E0002000: CID B105E00D PID 04000BB00B Gen FPB (type 0x0)
NXP: MKE06Z128xxx4
DAP stride is 1024 bytes (256 words)
Inspected v.2 On chip Kinetis Flash memory module FTMRE_KE06_04.cfx
Image 'FTMRE Kinetis 8K KE04 Tiny Jan 23 2020 14:57:48'
Opening flash driver FTMRE_KE06_04.cfx
Sending VECTRESET to run flash driver
Flash variant 'FTMRE Kinetis KE06_04' detected (128KB = 256*512 at 0x0)
Closing flash driver FTMRE_KE06_04.cfx
Connected: was_reset=true. was_stopped=true
```

```
Awaiting telnet connection to port 3332 ...
GDB nonstop mode enabled
Opening flash driver FTMRE_KE06_04.cfx (already resident)
Sending VECTRESET to run flash driver
Flash variant 'FTMRE Kinetis KE06_04' detected (128KB = 256*512 at 0x0)
Writing 12128 bytes to address 0x00000000 in Flash
00000200 done   4% (512 out of 12128)
(  8) at 00000200: 512 bytes - 1024/12128
00000600 done  12% (1536 out of 12128)
00000800 done  16% (2048 out of 12128)
00000A00 done  21% (2560 out of 12128)
00000C00 done  25% (3072 out of 12128)
00000E00 done  29% (3584 out of 12128)
00001000 done  33% (4096 out of 12128)
00001200 done  37% (4608 out of 12128)
00001400 done  42% (5120 out of 12128)
00001600 done  46% (5632 out of 12128)
00001800 done  50% (6144 out of 12128)
00001A00 done  54% (6656 out of 12128)
00001C00 done  59% (7168 out of 12128)
00001E00 done  63% (7680 out of 12128)
00002000 done  67% (8192 out of 12128)
00002200 done  71% (8704 out of 12128)
00002400 done  75% (9216 out of 12128)
00002600 done  80% (9728 out of 12128)
00002800 done  84% (10240 out of 12128)
00002A00 done  88% (10752 out of 12128)
00002C00 done  92% (11264 out of 12128)
00002E00 done  97% (11776 out of 12128)
00003000 done 100% (12288 out of 12128)
Sectors written: 23, unchanged: 1, total: 24
Erased/Wrote sector  0-23 with 12128 bytes in 1457msec
Closing flash driver FTMRE_KE06_04.cfx
Flash Write Done
Flash Program Summary: 12128 bytes in 1.46 seconds (8.13 KB/sec)
Starting execution using system reset and halt target
Stopped (Was Reset)  [Reset from Unknown]
Stopped: Breakpoint #1
```

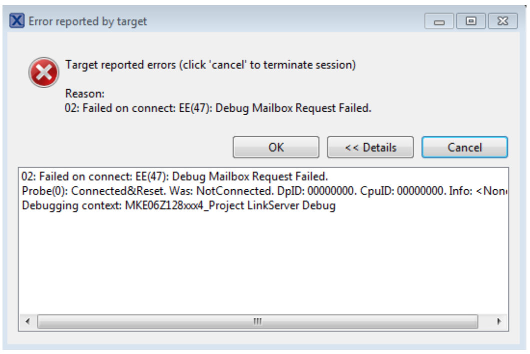
7 Recovering Secured MCUs

On occasion these MCUs may accidentally be programmed with ‘garbage’ data which can lead to a failure to establish a debug connection.

An MCU in this state will show a debug log similar to this:

```
MCUXpresso IDE RedlinkMulti Driver v11.1 (Feb  4 2020 08:54:29 - crt_emu_cm_red-
link build 9)
Found part description in XML file MKE06Z4_internal.xml
Reconnected to existing LinkServer process.
============= SCRIPT: ke0xconnect.scp =============
KE0X Connect Script
Probe Handle 1 Open
DpID = 0BC11477
Assert NRESET
Reset pin state: 00
Power up Debug
APID = 0x04770031
MDM-AP APID: 0x001C0020
Release NRESET
Reset pin state: 01
MDM-AP System Reset/Hold Reset/Debug Request
MDM-AP Control: 0x0000001C
MDM-AP Status (Flash Ready) : 0x00000006
Part is secured
Mass Erase Required
============= END SCRIPT ==========================
Probe Firmware: LPC-LINK2 CMSIS-DAP V5.361 (NXP Semiconductors)
Serial Number:  IQCYAWEV
VID:PID:  1FC9:0090
USB Path: \\?\hid#vid_1fc9&pid_0090&mi_00#8&149efaae&0&0000#{4d1e55b2-f16f-11cf-
88cb-001111000030}
Using memory from core 0 after searching for a good core
connection failed - EE(47): Debug Mailbox Request Failed... Retrying
probe 1 TAP 0 gives zero TAP ID - is probe connected to a target?
Using memory from core 0 after searching for a good core
Failed on connect: EE(47): Debug Mailbox Request Failed.
Connected&Reset. Was: NotConnected. DpID: 00000000. CpuID: 00000000. Info: <None>
Last stub error 0: OK
Last sticky error: 0x0 AIndex: 0
Debug bus selected: MemAp 0
DAP Speed test unexecuted or failed
Debug protocol: SWD. RTCK: Disabled. Vector catch: Disabled.
error closing down debug session - Nn(05). Wire ACK Fault in DAP access
```

And present various error dialogues to the user – such as:



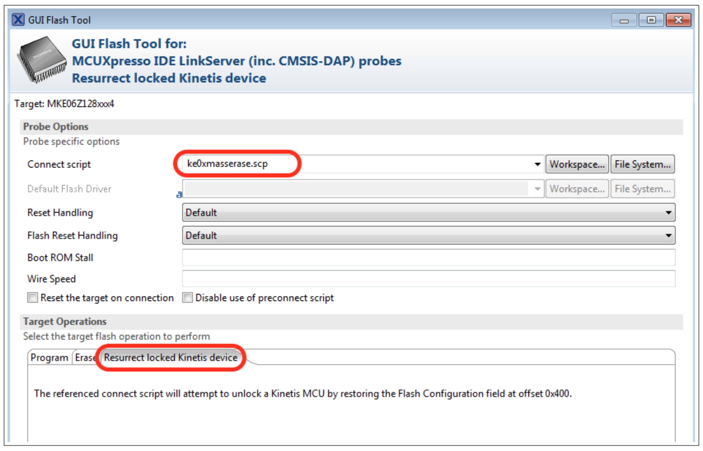
Target reported errors (click 'cancel' to terminate session)

Reason:
02: Failed on connect: EE(47): Debug Mailbox Request Failed.

```
02: Failed on connect: EE(47): Debug Mailbox Request Failed.
Probe(0): Connected&Reset. Was: NotConnected. DpID: 00000000. CpuID: 00000000. Info: <Non
Debugging context: MKE06Z128xxx4_Project LinkServer Debug
```

Should this occur the recovery method is as follows.

1. Power cycle the board
2. From within the IDE click the *'Clean Up Debug'* button to ensure all debug processes have been terminated
3. From within the *Project Explorer* view, select a project for the target board and then click the *'GUI Flash Tool'* button
4. After debug probe selection. click the *'Resurrect locked Kinetis device'* tab and …
5. Select the specific Connect script *'ke0xmasserase.scp'* from the drop down list. This recovery script is specific to these KE0x parts.


GUI Flash Tool

GUI Flash Tool for:
MCUXpresso IDE LinkServer (inc. CMSIS-DAP) probes
Resurrect locked Kinetis device

Target: MKE06Z128xxx4

Probe Options
Probe specific options

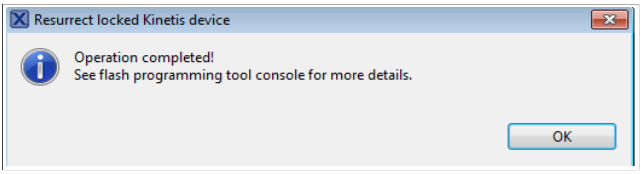
Connect script: ke0xmasserase.scp
Default Flash Driver:
Reset Handling: Default
Flash Reset Handling: Default
Boot ROM Stall:
Wire Speed:
☐ Reset the target on connection ☐ Disable use of preconnect script

Target Operations
Select the target flash operation to perform

Program | Erase | Resurrect locked Kinetis device

The referenced connect script will attempt to unlock a Kinetis MCU by restoring the Flash Configuration field at offset 0x400.

6. Click *'Run...'*
7. A successful recovery will generate a dialogue as below:


Resurrect locked Kinetis device

Operation completed!
See flash programming tool console for more details.

The part is now recovered and future debug sessions should now occur as normal.