# AJSM Lock and Unlock

**November 2017**
**QJ Wang**

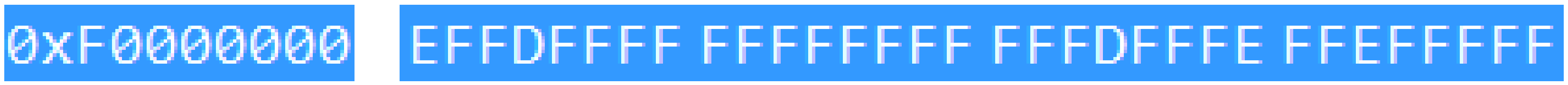
Texas Instruments

# Advanced JTAG Security Module (AJSM) module

This device includes a an Advanced JTAG Security Module (AJSM) module. The AJSM provides security to the memory content of the device by letting users secure the device after programming.

**The AJSM has four features:**

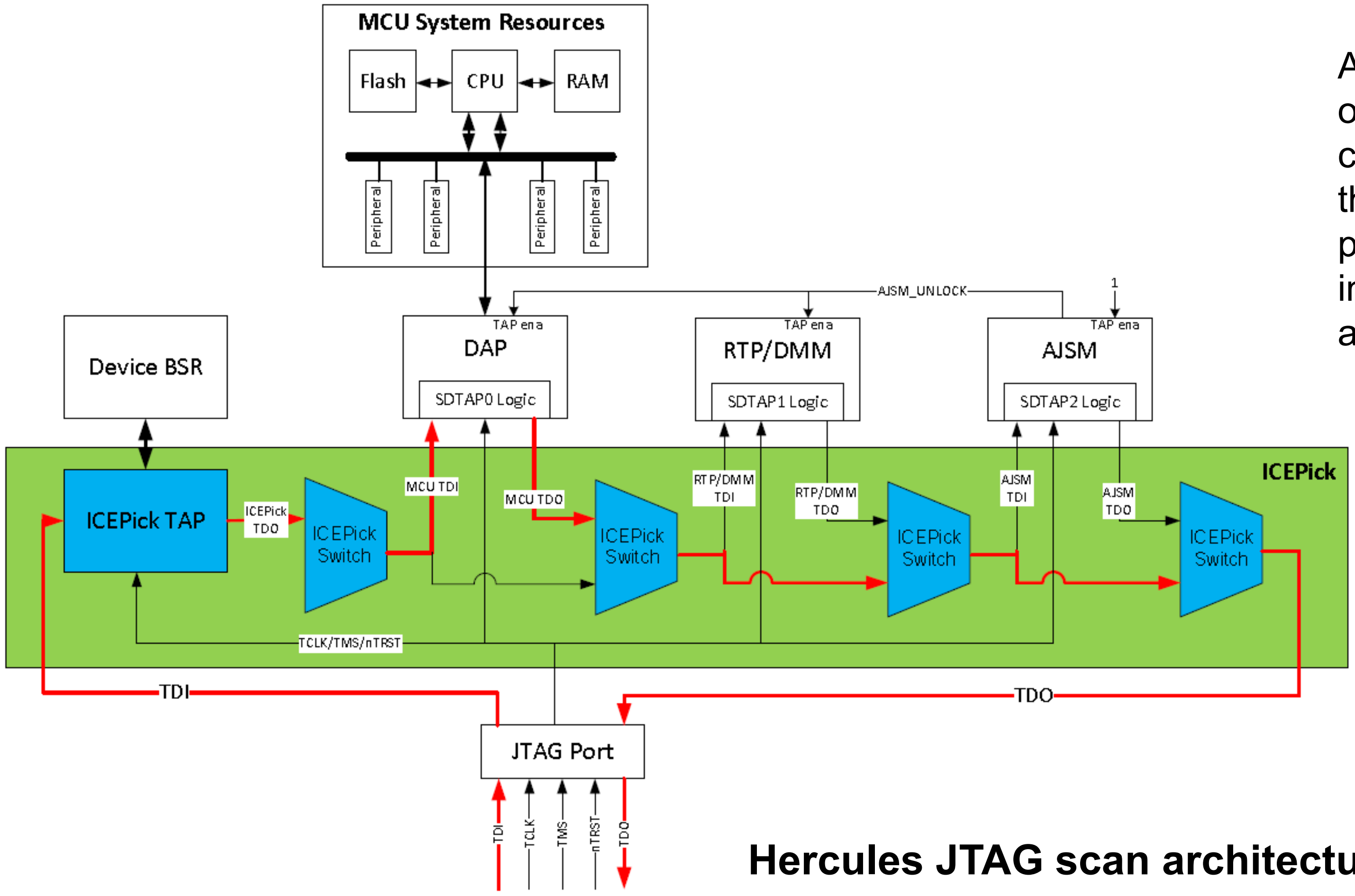
1. a TI programed visible unlock code,
2. a customer defined 128-bit key protected by ECC,
3. a scan path for unlocking the JTAG scan chain,
4. a key for permanently locking a device.



```
0xF0000000    EFFDFFFF FFFFFFFF FFFDFFFE FFEFFFFF
```

The devices shipped from TI are programmed with the visible unlock code, any other value will result in locking a device. You can lock the device by programming any number of bits and your ECC. If all the bits are programmed (0), the device will be permanently locked.

# Advanced JTAG Security Module (AJSM) module



**Hercules JTAG scan architecture**

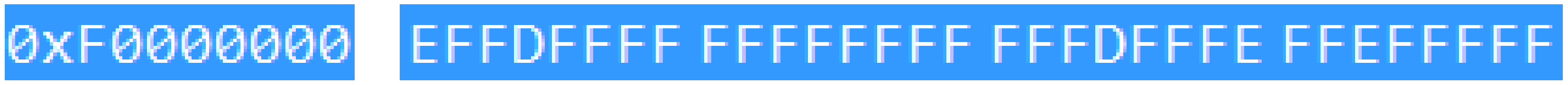
After reset, the ICEPick TAP is the only TAP connected in the JTAG scan chain. All other TAPs are classified as the Secondary Debug TAPs. From the perspective of the external JTAG interface, **secondary debug TAPs** appear to not exist after reset.

When the device is in a secured state, the only secondary debug TAP that can be put in the JTAG scan chain is the AJSM TAP. All other TAPs, whether they are secondary debug TAPs or test TAPs, are disabled from access

## AJSM Key Generation

The AJSM key is stored in memory that only allows changing 1s to 0s.

TI ships the devices programmed with the 128-bit visible unlock code at address 0xF00000000 and the ECC at address of 0xF0040000.
TI Visible Unlock Code (TMS570LC43x) at 0xF00000000 (customer OTP bank 0):



```
0xF0000000   EFFDFFFF FFFFFFFF FFFDFFFE FFEFFFFF
```

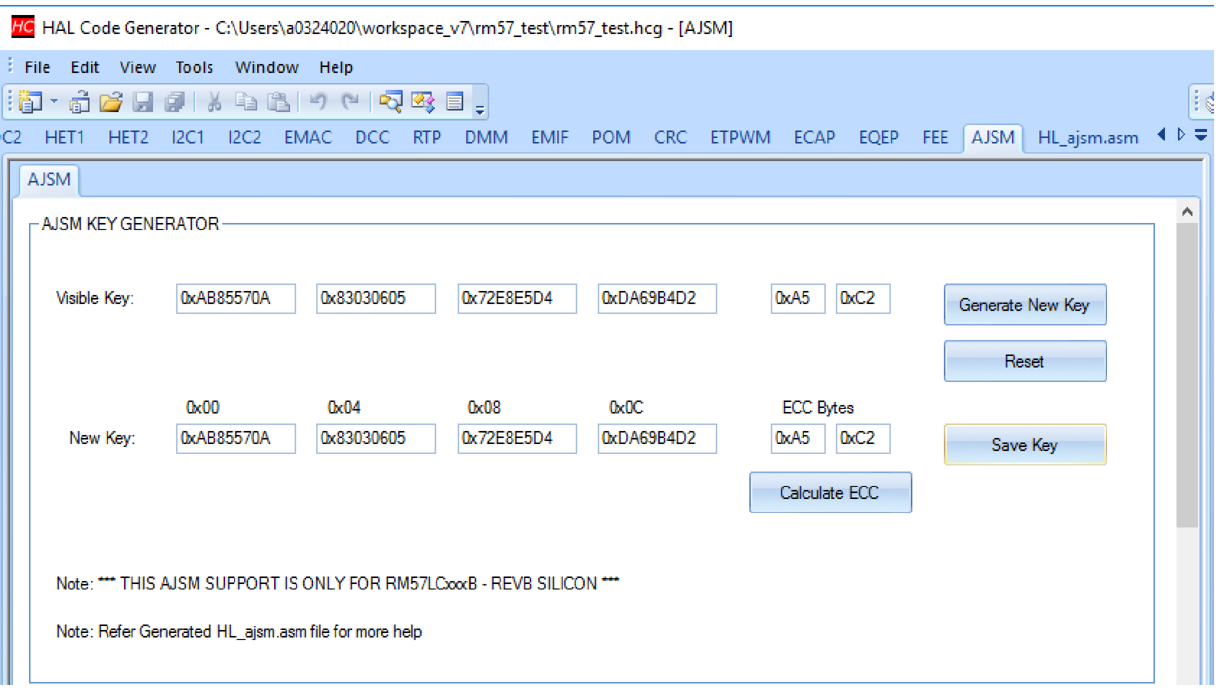
ECC: 0xEDED

This code is mainly 1s to allow for a custom key effective to 123 bits. The ECC is calculated based on the device's endianness.

## AJSM Locking

- The F021 Flash API supports programming the customer OTP.
- TI and third party tools support programming the customer OTP either integrated into the application or standalone. Standalone programming of the AJSM is recommended for production
- Once a device is locked, the ICEPICK and AJSM will be the only visible tap (Test Access Port); the scan chain will still be available and discoverable as a length of 6 to allow for easy identification that the device is functional. When the AJSM is locked, the typical emulator response will be no communication.
- To differentiate JTAG communication issue from a locked AJSM, perform a scan chain check or read the JTAG ID.

# AJSM Key Generation


HAL Code Generator - C:\Users\a0324020\workspace_v7\rm57_test\rm57_test.hcg - [AJSM]

File Edit View Tools Window Help
C2 HET1 HET2 I2C1 I2C2 EMAC DCC RTP DMM EMIF POM CRC ETPWM ECAP EQEP FEE AJSM HL_ajsm.asm
AJSM
AJSM KEY GENERATOR
Generate New Key
Reset
Save Key
Calculate ECC

| | 0x00 | 0x04 | 0x08 | 0x0C | ECC Bytes | |
|---|---|---|---|---|---|---|
| Visible Key: | 0xAB85570A | 0x83030605 | 0x72E8E5D4 | 0xDA69B4D2 | 0xA5 | 0xC2 |
| New Key: | 0xAB85570A | 0x83030605 | 0x72E8E5D4 | 0xDA69B4D2 | 0xA5 | 0xC2 |

Note: *** THIS AJSM SUPPORT IS ONLY FOR RM57LCxxxB - REVB SILICON ***

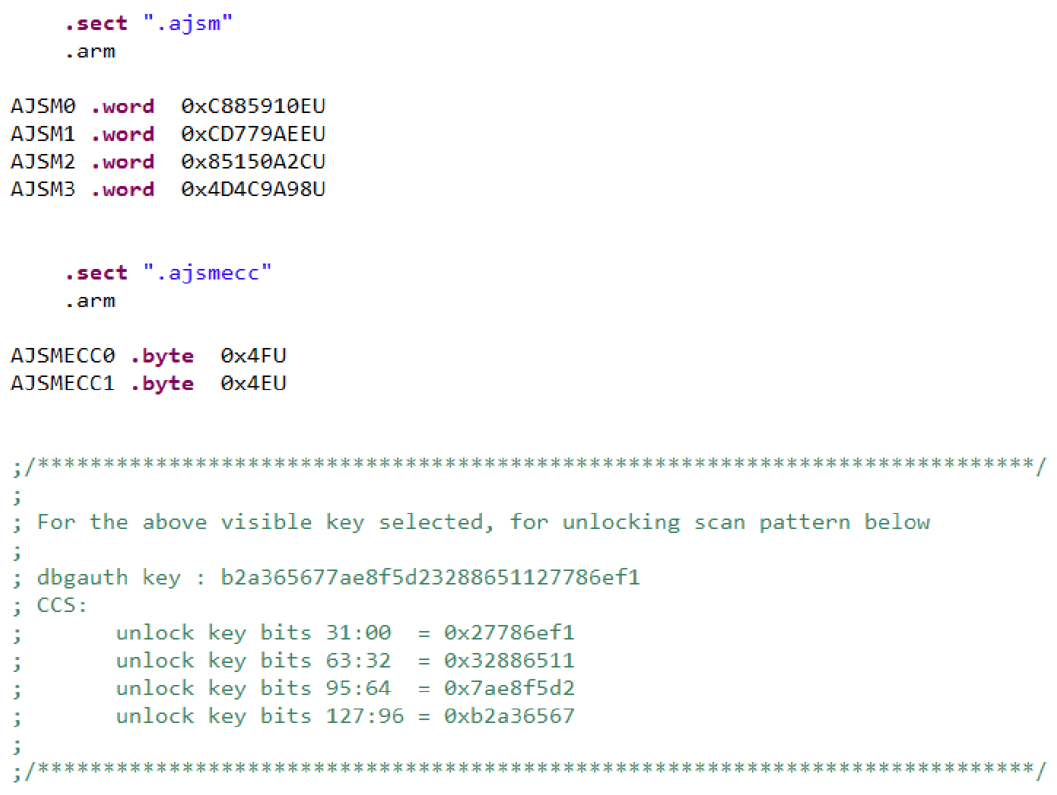
Note: Refer Generated HL_ajsm.asm file for more help

The following steps have to be followed to generate AJSM Keys:

1. Open a HALCoGen Project with HALCoGen Version 4.06.00 or Greater.
2. Go to the “Driver Enable” tab and check 'Enable AJSM'.
3. 2 Options:
   - HALCoGen “Generate Key” AJSM Unlock Key:
   - Manually Feed Key:

   Once satisfied with the Key, choose “Save Key”. Notice the “New Key” gets saved to the “Visible Key”.
4. Generate the code. File “ajsm.asm” gets generated, which contains the necessary memory or data section with the new AJSM keys. This new ajsm keys are stored in HALCoGen project and when next time same project is opened this key will appears in the “Visible Key” Field.
5. Memory section must be added to the linker file to program the "ajsm" data section to address location 0xF0000000 and “ajsmecc” data section to address location 0xF0400000, before building the project.

# ajsm.asm Generated through HALCoGen

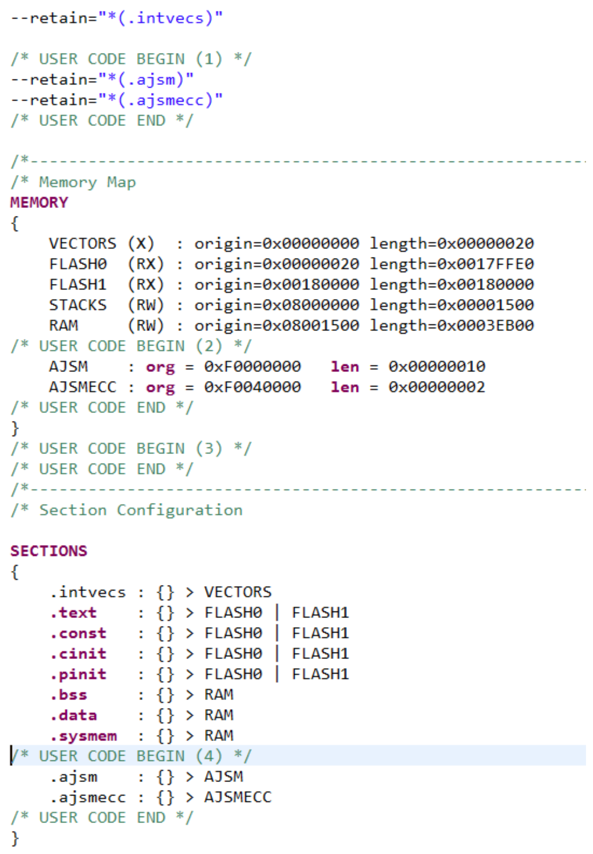


```
    .sect ".ajsm"
    .arm

AJSM0 .word  0xC885910EU
AJSM1 .word  0xCD779AEEU
AJSM2 .word  0x85150A2CU
AJSM3 .word  0x4D4C9A98U


    .sect ".ajsmecc"
    .arm

AJSMECC0 .byte  0x4FU
AJSMECC1 .byte  0x4EU


;/*****************************************************************************/
;
; For the above visible key selected, for unlocking scan pattern below
;
; dbgauth key : b2a365677ae8f5d23288651127786ef1
; CCS:
;       unlock key bits 31:00  = 0x27786ef1
;       unlock key bits 63:32  = 0x32886511
;       unlock key bits 95:64  = 0x7ae8f5d2
;       unlock key bits 127:96 = 0xb2a36567
;
;/*****************************************************************************/
```

- User must take necessary precautions to save this project file securely. The HALCoGen project file (*.dil) contains the new Key information in plain txt.

- If you have already programmed the Keys without using HALCoGen, the HALCoGen project will still have the default key. It is the user's responsibility to manually feed the current Keys using the "New Key" fields and Save it to reflect in the "Visible Key".

# Link.cmd generated through HALCoGen

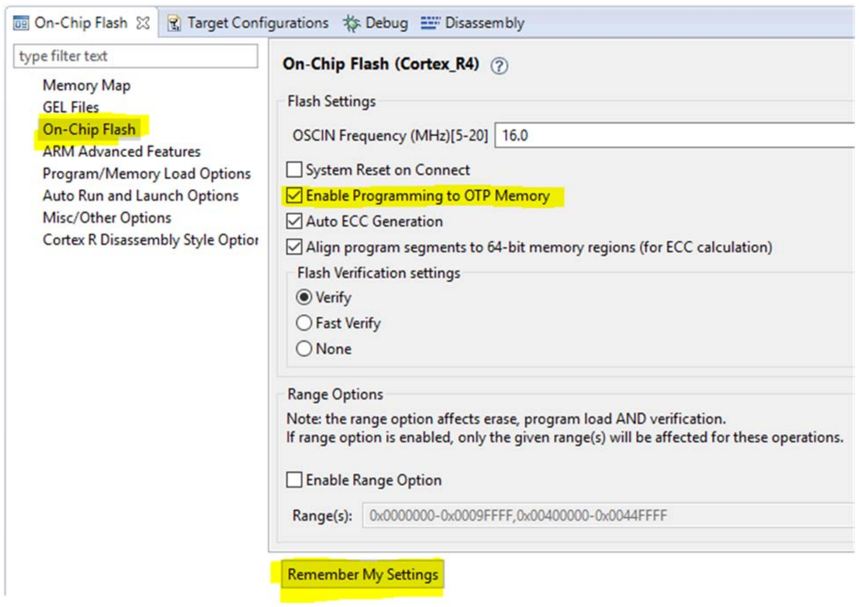


```
--retain="*(.intvecs)"

/* USER CODE BEGIN (1) */
--retain="*(.ajsm)"
--retain="*(.ajsmecc)"
/* USER CODE END */

/*----------------------------------------------------------------------------
/* Memory Map
MEMORY
{
    VECTORS (X)  : origin=0x00000000 length=0x00000020
    FLASH0  (RX) : origin=0x00000020 length=0x0017FFE0
    FLASH1  (RX) : origin=0x00180000 length=0x00180000
    STACKS  (RW) : origin=0x08000000 length=0x00001500
    RAM     (RW) : origin=0x08001500 length=0x0003EB00
/* USER CODE BEGIN (2) */
    AJSM    : org = 0xF0000000   len = 0x00000010
    AJSMECC : org = 0xF0040000   len = 0x00000002
/* USER CODE END */
}
/* USER CODE BEGIN (3) */
/* USER CODE END */
/*----------------------------------------------------------------------------
/* Section Configuration

SECTIONS
{
    .intvecs : {} > VECTORS
    .text    : {} > FLASH0 | FLASH1
    .const   : {} > FLASH0 | FLASH1
    .cinit   : {} > FLASH0 | FLASH1
    .pinit   : {} > FLASH0 | FLASH1
    .bss     : {} > RAM
    .data    : {} > RAM
    .sysmem  : {} > RAM
/* USER CODE BEGIN (4) */
    .ajsm    : {} > AJSM
    .ajsmecc : {} > AJSMECC
/* USER CODE END */
}
```

1. Add Memory Section in linker file to program the ""ajsm"" data section to address location 0xF000000.
2. Add linker setting
   --retain="*(.ajsm)“
   --retain="*(.ajsmecc)"
3. Build the project and load the code
4. Once device programmed correctly, after reset your JTAG will be locked.

## Program Key to OTP

- Launch the target configuration, and connect JTAG to the device
- Open CCSàToolsàOn-Chip Flash, CHECK “Enable Programming to OTP Memory”, then Click “Remember My Settings”
- Load program
- Once device programmed correctly, after reset your JTAG will be locked.



On-Chip Flash (Cortex_R4)

Flash Settings

OSCIN Frequency (MHz)[5-20] 16.0

- [ ] System Reset on Connect
- [x] Enable Programming to OTP Memory
- [x] Auto ECC Generation
- [x] Align program segments to 64-bit memory regions (for ECC calculation)

Flash Verification settings

● Verify
○ Fast Verify
○ None

Range Options

Note: the range option affects erase, program load AND verification.
If range option is enabled, only the given range(s) will be affected for these operations.

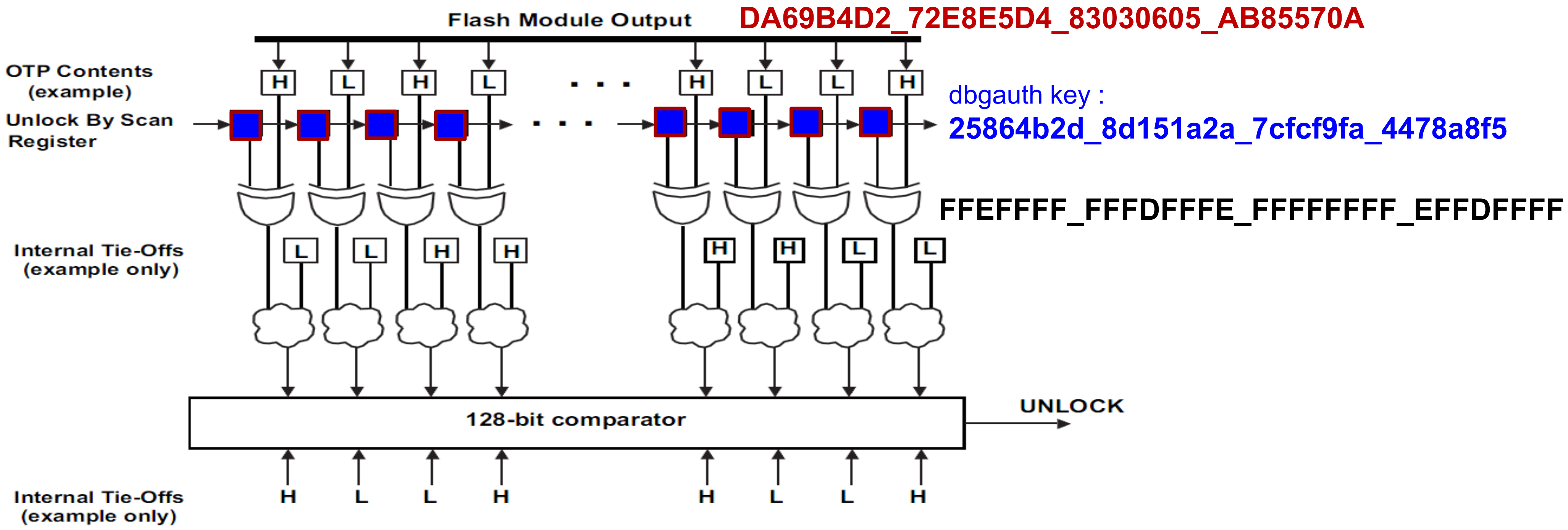
- [ ] Enable Range Option

Range(s): 0x0000000-0x0009FFFF,0x00400000-0x0044FFFF

## AJSM Temporarily Unlocking

0xF0000000 EFFDFFFF FFFFFFFF FFFDFFFE FFEFFFFF

The Hercules AJSM can be temporarily unlocked by scanning an appropriate value into the "Unlock By Scan" register of the AJSM module. The XOR of the OTP contents and the Unlock-By-Scan register contents results in the original visible unlock code. The device will remain unlocked through a system reset, but not power on reset.



Flash Module Output
DA69B4D2_72E8E5D4_83030605_AB85570A
OTP Contents (example)
Unlock By Scan Register
dbgauth key :
25864b2d_8d151a2a_7cfcf9fa_4478a8f5
FFEFFFFF_FFFDFFFE_FFFFFFFF_EFFDFFFF
Internal Tie-Offs (example only)
128-bit comparator
UNLOCK
Internal Tie-Offs (example only)
H L H L H L L H
L L H H H H L L
H L L H H L L H

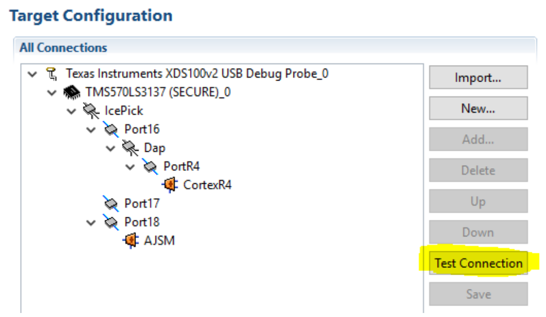
Texas Instruments

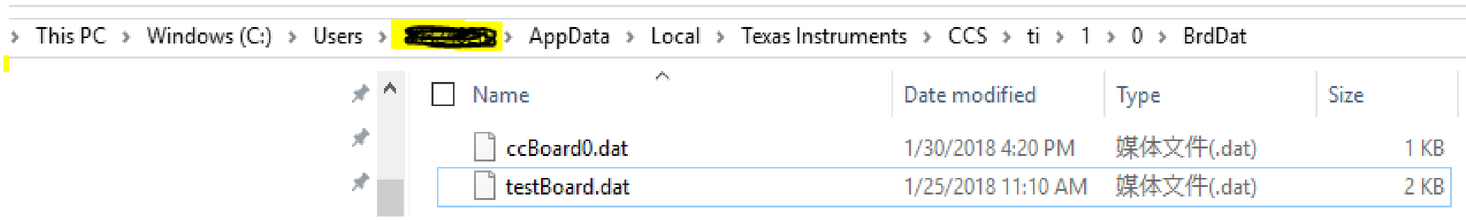
# AJSM Temporarily Unlocking -- Command Line

**Dbgauth** facilitates debugger authentication on supported TI SOC platforms. It is located in: CCSv7 - \ccs_base\common\uscif . ***Example:***

***dbgauth -c testBoard.dat -s ajsm -t cortexr4 -k 25864b2d8d151a2a7cfcf9fa4478a8f5 -m 1***

1. testBoard.dat generation:
   In CCS target configuration window, click the **Test Connection** button. A test connection window will pop up and list the path to the generated board.dat file. The default path is C:\users/<account name>\AppData\Local \TEXAS~1\CCS\ti\<#>\<#>\BrdDat\testboard.dat.




This PC › Windows (C:) › Users › [redacted] › AppData › Local › Texas Instruments › CCS › ti › 1 › 0 › BrdDat

| Name | Date modified | Type | Size |
|---|---|---|---|
| ccBoard0.dat | 1/30/2018 4:20 PM | 媒体文件(.dat) | 1 KB |
| testBoard.dat | 1/25/2018 11:10 AM | 媒体文件(.dat) | 2 KB |

2. The unlock key is generated through HALCoGen and located in ajsm.asm file

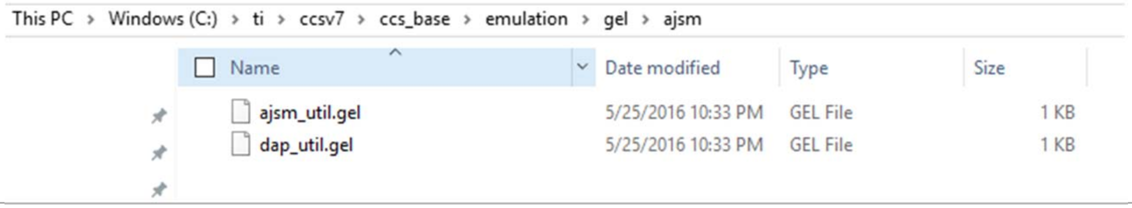
# AJSM Temporarily Unlocking -- CCS

The dbgauth tool can be integrated into CCS using GEL as a means of automatically unlocking the device as part of the startup.

1. Open the gel file for your device. The original gel file is located at ***C:\ti\ccsv7\ccs_base\emulation\gel\***
2. Add the following **blue section** to gel file (modify the folder name for your testBoard.dat, and the unlock key)

```
/*----------------------------------------------------------------------------*/
/* Function - StartUp()                                                        */
StartUp(){
   AJSM_Unlock_Demo();
} /* StartUp() */

/* StartUp() */ /*------------------------------------------------------------------------------*/
menuitem "TMS570LS0714 AJSM Unlock Demo";
hotmenu AJSM_Unlock_Demo(){
  GEL_System("C:\\ti\\ccsv7\\ccs_base\\common\\uscif\\dbgauth.exe –c c:\\Users\\??\\AppData\\Local\\TEXASI~1\\CCS\\ti\\1\\0
\\BrdDat\\testBoard.dat -s ajsm -t cortexr4 -k b2a365677ae8f5d23288651127786ef1 -m 1");
}
```

3. Copy the ajsm folder to ***C:\ti\ccsv7\ccs_base\emulation\gel\*** (ajsm folder is from SPNA232.zip)

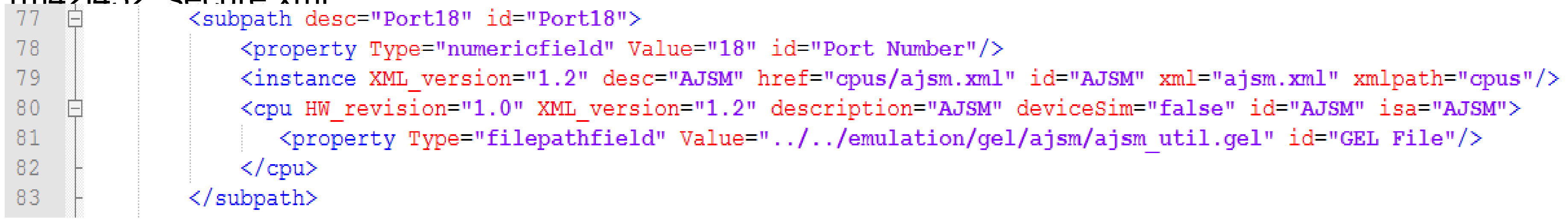

This PC › Windows (C:) › ti › ccsv7 › ccs_base › emulation › gel › ajsm

| Name | Date modified | Type | Size |
|---|---|---|---|
| ajsm_util.gel | 5/25/2016 10:33 PM | GEL File | 1 KB |
| dap_util.gel | 5/25/2016 10:33 PM | GEL File | 1 KB |

# AJSM Temporarily Unlocking -- CCS

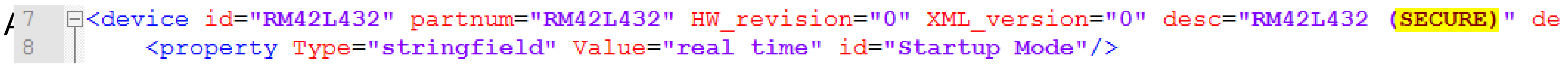
4. In the attached zip file, there are new versions of devices with “_secure” that can be selected when creating a target configuration. For example TMS570LS3137_secure.xml. The xml files are located in C:\ti\ccsv7\ccs_base\common\targetdb\devices\

4. Open the xml file for your device (for example rm42l432.xml), add following lines to the xml file and save it as rm42l432_secure.xml



```
        <subpath desc="Port18" id="Port18">
            <property Type="numericfield" Value="18" id="Port Number"/>
            <instance XML_version="1.2" desc="AJSM" href="cpus/ajsm.xml" id="AJSM" xml="ajsm.xml" xmlpath="cpus"/>
            <cpu HW_revision="1.0" XML_version="1.2" description="AJSM" deviceSim="false" id="AJSM" isa="AJSM">
                <property Type="filepathfield" Value="../../emulation/gel/ajsm/ajsm_util.gel" id="GEL File"/>
            </cpu>
        </subpath>
```

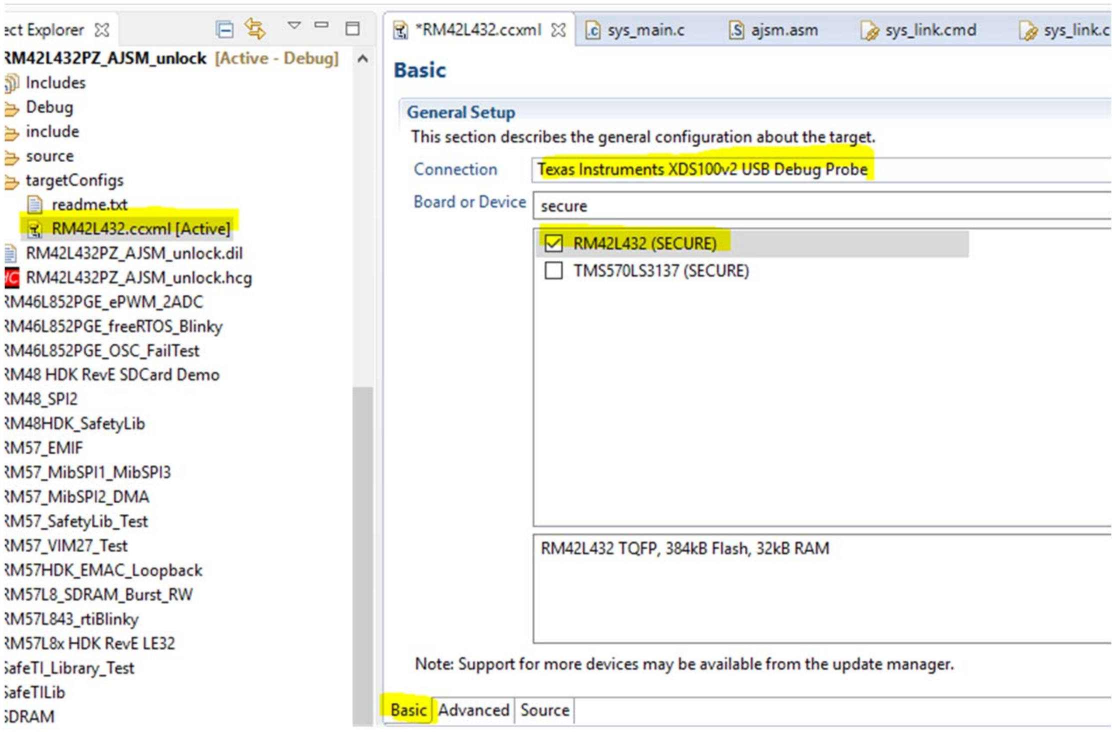
A



```
<device id="RM42L432" partnum="RM42L432" HW_revision="0" XML_version="0" desc="RM42L432 (SECURE)" de
    <property Type="stringfield" Value="real time" id="Startup Mode"/>
```

# AJSM Temporarily Unlocking -- CCS

6. Open the target configuration window, Select the new device xml file (with secure)



This section describes the general configuration about the target.

Connection: Texas Instruments XDS100v2 USB Debug Probe

Board or Device: secure

[✓] RM42L432 (SECURE)

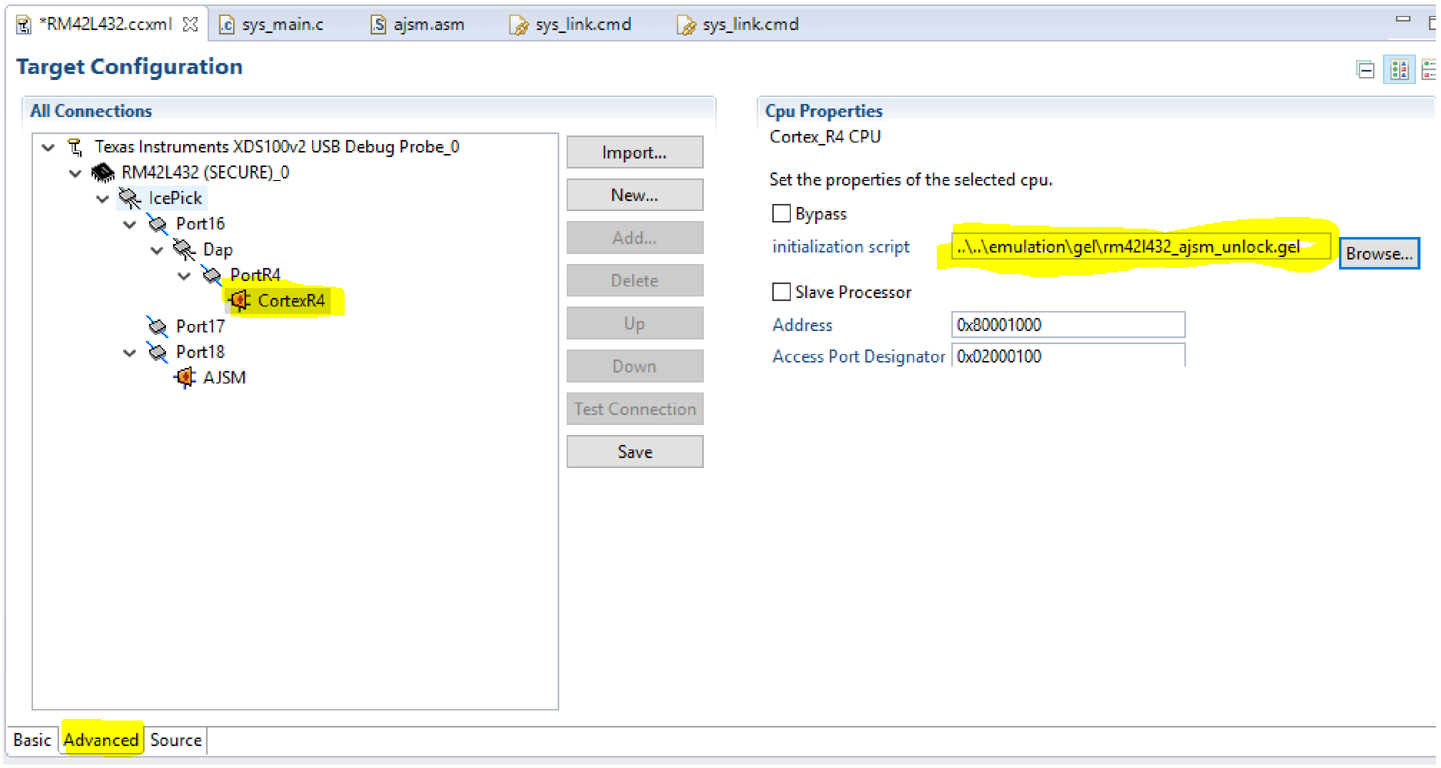
[ ] TMS570LS3137 (SECURE)

RM42L432 TQFP, 384kB Flash, 32kB RAM

Note: Support for more devices may be available from the update manager.

# AJSM Temporarily Unlocking -- CCS

7. Click Advanced, click CortexR4, and select rm42l432_ajsm_unlock.gel as the initialization script



Cpu Properties

Cortex_R4 CPU

Set the properties of the selected cpu.

- [ ] Bypass

initialization script: ..\..\emulation\gel\rm42l432_ajsm_unlock.gel

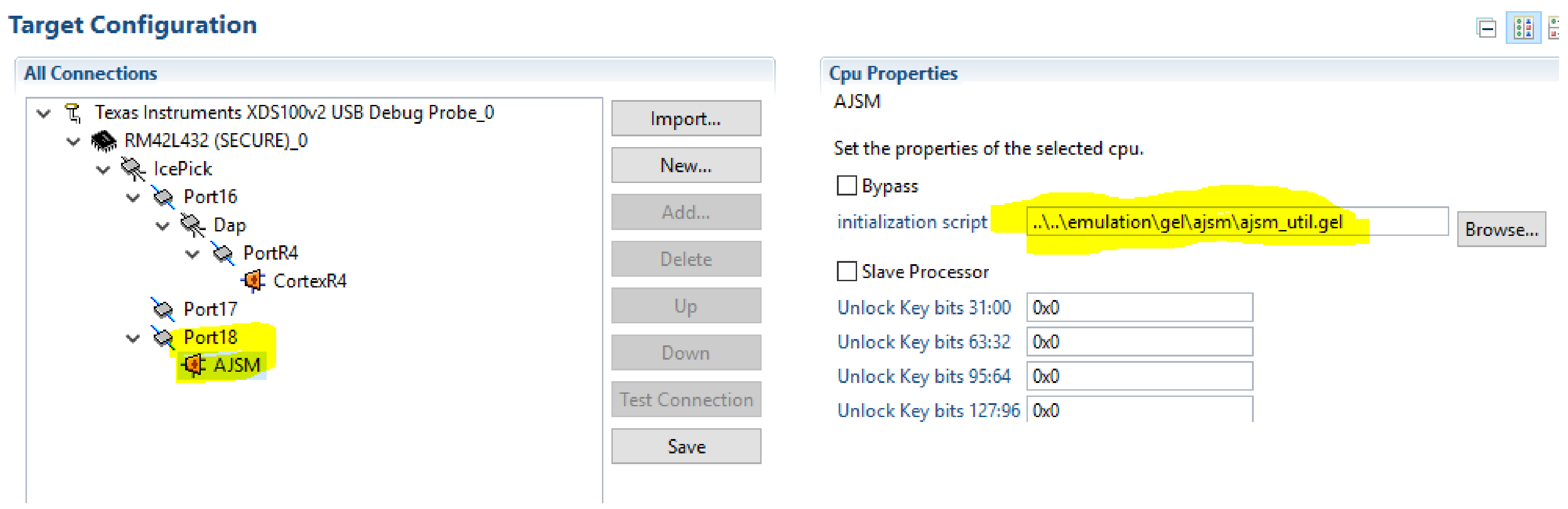
- [ ] Slave Processor

Address: 0x80001000

Access Port Designator: 0x02000100

## AJSM Temporarily Unlocking -- CCS

8. Make the settings for Port18-AJSM are correct



Target Configuration

All Connections

- Texas Instruments XDS100v2 USB Debug Probe_0
  - RM42L432 (SECURE)_0
    - IcePick
      - Port16
        - Dap
          - PortR4
            - CortexR4
      - Port17
      - Port18
        - AJSM

Cpu Properties

AJSM

Set the properties of the selected cpu.

☐ Bypass

initialization script: ..\..\emulation\gel\ajsm\ajsm_util.gel

☐ Slave Processor

| | |
|---|---|
| Unlock Key bits 31:00 | 0x0 |
| Unlock Key bits 63:32 | 0x0 |
| Unlock Key bits 95:64 | 0x0 |
| Unlock Key bits 127:96 | 0x0 |

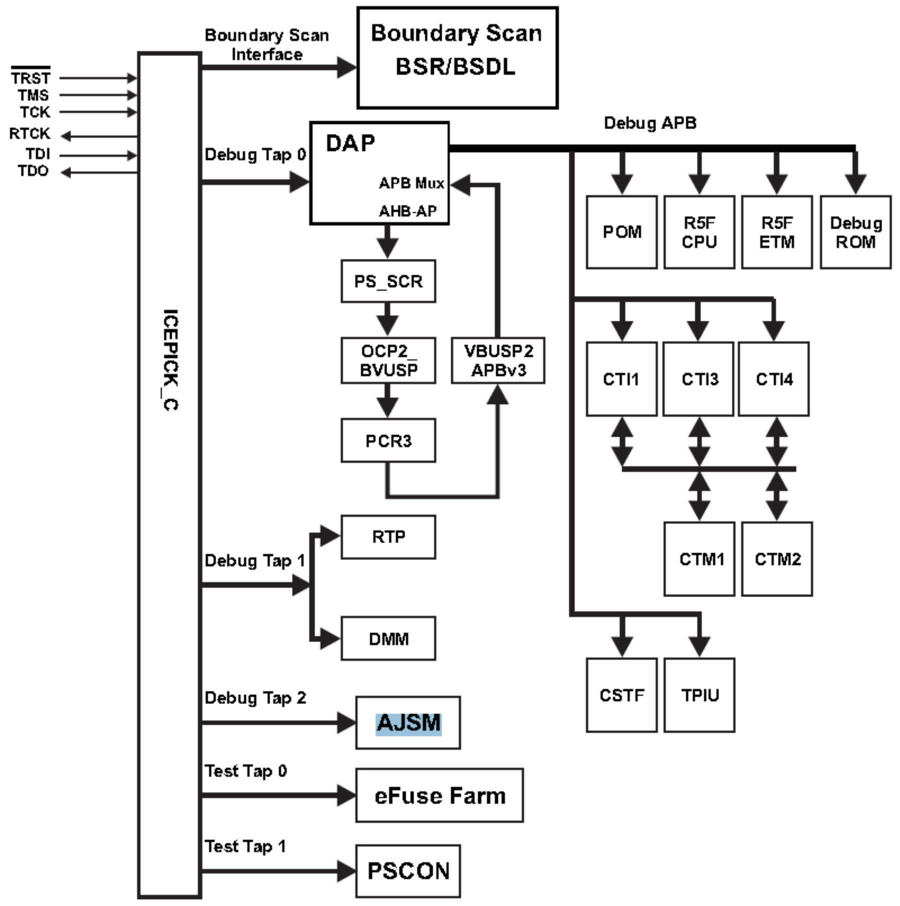
9. Then launch rm42l432.ccxml target configuration file, click connect, done

## AJSM Temporarily Unlocking -- CCS

Extract the files in spna232.zip to the device folder of code composer studio. The standard path is C:\ti\ccsv7\ccs_base\common\targetdb\devices\. If the desired part number does not exist, follow the instructions in the readme.txt file to generate it.

http://www.ti.com/general/docs/lit/getliterature.tsp?baseLiteratureNumber=spna232&fileType=zip

# AJSM: Connection Block Diagram



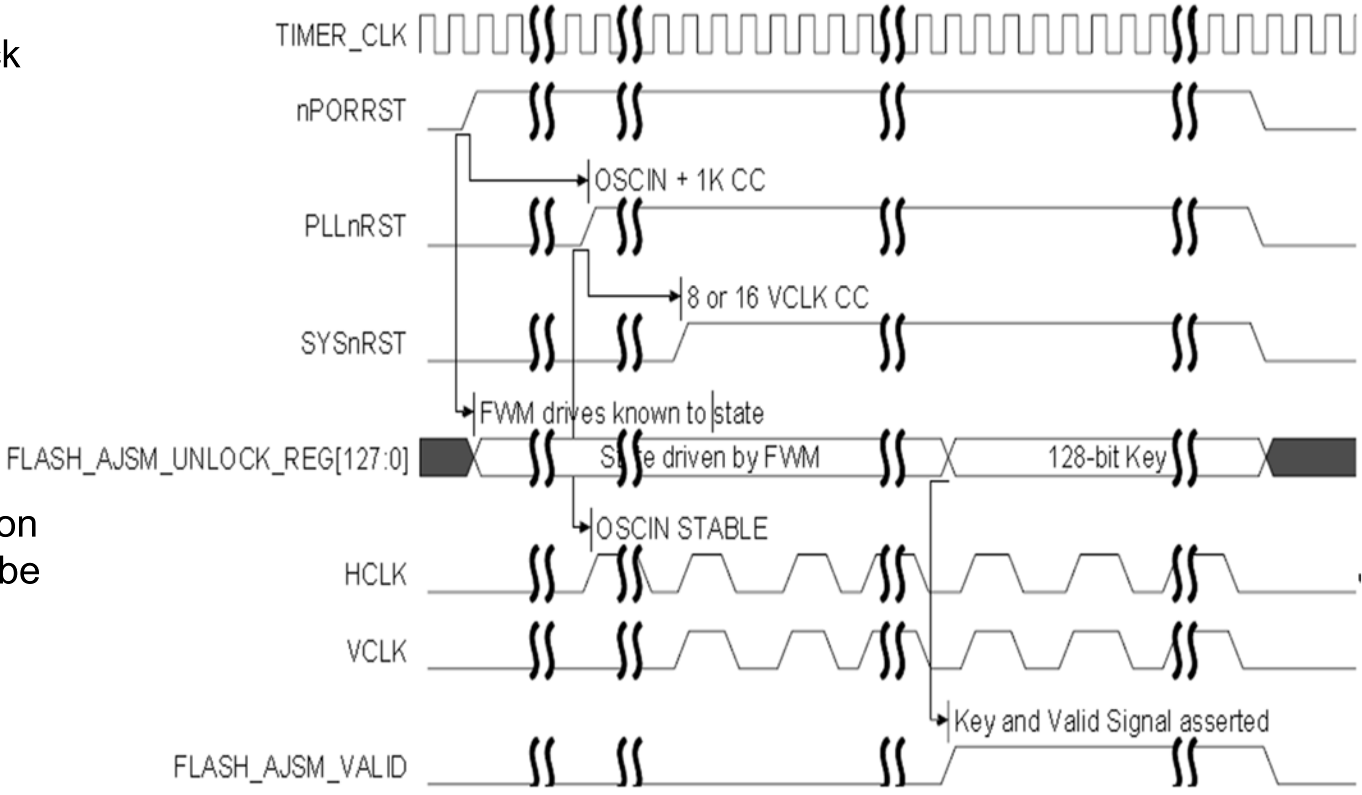
- A secure device only permits JTAG accesses to the AJSM scan chain via the Secondary Tap # 2 of the ICEPick module.
- All other secondary taps, test taps and the boundary scan interface are not accessible in this state.

# AJSM Visible Unlock Code

The flash wrapper drives the visible unlock code.

When the flash pump is powering up, the wrapper drives all ones to this location. In addition, it drives FLASH_AJSM_VALID signal low which indicates that the visible unlock code is not ready to be read.

When the flash is ready, the first read access is to the visible unlock code location and the FLASH_AJSM_VALID signal will be high. The FLASH_AJSM_VALID signal is level triggered.



## AJSM Emulation

***Emulation:***
Emulators and programmers can scan the JTAG IDCODE register to determine if a device is secure or unsecure.

If the IDCODE of the AJSM TAP is returned, the device is secure. The tool must then scan the proper value into the “Unlock by scan” register. It can then try to read the IDCODE of other STAPs or TEST TAPs. Unless the device was secured with all ones or all zeros in the visible unlock code location, or the “unlock by scan” value was incorrect, the tool should be able to access other TAP’s on the ICEPICK.

***Maintaining the Unlock Condition during Emulation***
Before an emulator is able to access a device, the device must be allowed to go through a normal reset sequence, to allow the device to load the visible unlock code into the UNLOCK_REG register. The “unlock by scan register” is reset by nPORRST.