uTasker Bootloader

Emphasis on IP and cloning protection. Supports:

- Encrypted user application (proprietary format AES encryption)
- Encrypted (RT1024 internal) SPI flash content, with XIP On-The-Fly decryption from unreadable hidden key generated during bootloader installation process and stored within CPU (not QSPI flash).

SW-GP2 protected AES256 key inside processor, only readable by BEE AES128 Decryption module for OTF decryption. “aes256 secret key” is default key used in builds until changed.

Implemented as 3 separate sequentially-staged boot-loaders. The first two stages are built, then combined and installed as a single image file. That loads 3rd stage. Also built: combined 3-stages (only load one image).

Loader Stage	Encryption	Target	Function	Image
1. Bare Minimum	None	1	boot configuration (SPI type, read by the ROM loader at reset) at the start of SPI flash. Try launch application or serial boot-loader.	TaskerBootLoaderImage.bin
2. Fall-back Serial	AES256	2		
2. Serial		3	Main boot-loader...	

Documents

https://www.utasker.com/docs/uTasker//uTaskerSerialLoader.pdf Overview of loader without specifics for iMX.RT family.

https://www.utasker.com/docs/iMX/i.MX_RT_1024_uTasker.pdf uTasker hardware setup and drivers for 1024, says see 1021 version… https://www.utasker.com/docs/iMX/i.MX_RT_1021_uTasker.pdf.

uTasker Bootloader Installation, Configuration, and Test

1. Cloned Mark's repository 21-Feb-2024.
2. Selected and checked out 2.0 branch

Per https://www.utasker.com/docs/iMX/MCUXpresso.pdf:

1. copied MCUxpresso/iMX project settings into root of repository
2. Imported project into MCUxpresso (upgraded to 11.9 and SDK 2.15 first)
3. verified build variables set correctly for all build configurations (RT1024 etc, no changes were required)
4. set active configuration to "uTasker Boot for XiP"
5. verified compiler settings (architecture M7 etc)
6. verified correct linker script set
7. verified this application builds.
8. Build all 4 (see below) and verify everything builds AOK.

Boot-loader and boot-loadable application must be built sequentially using these active build configurations:

1. `uTaskerBoot (uTasker Boot for XiP)`
2. `uTaskerFallbackLoader`
3. `uTaskerSerialBoot`
4. `5. uTaskerV1.4_BM_XIP` (uTasker example to be loaded via serial loader – probably not helpful here)

Create a test blinky application test per instructions in https://www.utasker.com/docs/iMX/MCUXpresso.pdf:

1. For LED, changed default to GPIO3 IO 30 (update `pin_mux.h` definitions and `pin_mux.c` initialization)
2. Verified blinky actually blinks on board AOK (before changes to make blinky boot-loadable).
3. Set `0x60020400` flash start address
4. Add post-build step "${ProjDirPath}/generate.bat" "${BuildArtifactFileBaseName}", plus generate.bat and boot_header.txt per instructions.
5. Verified assorted .bin files built AOK.

Update hardware MCU configuration in `Applications\uTaskerBoot\config.h`:

1. Correct the target MCU: undefine MIMXRT1060, define MIMXRT1024.
2. Comment out BOOT_LOADER_SUPPORTS_SDRAM as SensorBox has no SDRAM.
3. Rebuild all 4 targets in order – AOK.

Serial Loader User's Guide configuration:

1. `uTaskerSerialBoot\app_hw_iMX.h`:
   Dangerous! Multiple `app_hw_iMX.h` files (in uTaskerV1.4 + SerialLoader projects)
   This file has basic RT1024 definitions (clock configuration etc.).
   Appears to have pin definitions for EVK ie board control of PHY (for what board not documented).
   1. `#define USER_LED (PIN_GPIO_AD_B1_08_GPIO1_IO24)` ie `PORT1_BIT24` changed to:
      `#define USER_LED (PIN_GPIO_SD_B1_10_GPIO3_IO30)` ie `PORT3_BIT30`
   2. update LED init: `#define INIT_WATCHDOG_LED() _CONFIG_DRIVE_PORT_OUTPUT_VALUE(3 /*DRN: port was 1*/, (BLINK_LED), (BLINK_LED), (PORT_SRE_SLOW | PORT_DSE_HIGH))`
   3. update LED toggle:
      `#define TOGGLE_WATCHDOG_LED() _TOGGLE_PORT(3 /* DRN: was port 1, cannot use macro here */, BLINK_LED)`
   4. Defines `USER_BUTTON PIN_WAKEUP_GPIO5_IO00` - SensorBox has no user button – disable how?
      The WakeUp pin 52 is not connected in SensorBox – should be OK.
   5. Per Mark Butchers instructions, as there is no user button, set:
      ```
      #define FORCE_BOOT() 1 // DRN: always start serial loader
      #define RETAIN_LOADER_MODE() 0 // DRN: never stay in serial loader after
      checking application
      ```
   6. ToDo: Implement user-status notification via macros `#define _DISPLAY_xxx`

2. `uTaskerSerialBoot\config.h`:
   Dangerous!! Multiple `config.h` files (in uTaskerBoot, SerialLoader, and uTaskerV1.4 projects)
   Set up for USB stick (host mode) and no other options, lines ~1826-1831:
   1. `//#define USB_MSD_DEVICE_LOADER    // DRN: disabled // USB-MSD device mode`
      `#define USB_MSD_HOST_LOADER       // DRN: enabled  // USB-MSD host mode`
   2. Ethernet interface is already disabled in line ~1831: `//#define ETH_INTERFACE`
   3. How to ensure other loader modes disabled? (RS485, MODBUS, SREC…)??
   4. per Mark Butcher: `#define USB_MSD_HOST_TIMEOUT // DRN: enabled`
3. `uTaskerSerialBoot\Loader.h`:
   1. line 324: `#define NEW_SOFTWARE_FILE "software.bin"` left as-is for now

Test!
1. Load board with `uTaskerBootComplete_MIMXRT1024.bin` (using Segger Jlink loader)
2. LED is lit as expected (note I changed `INIT_WATCHDOG_LED()` to initially turn LED on).
3. Copied `evkmimxrt1024_iled_blinky_XiP.bin` to USB stick and renamed to `software.bin`
4. Inserted USB stick. Stick LED lights, briefly blinks off then steady on.
   LED remains on so 'blinky' application is not running.
5. Tried again with `evkmimxrt1024_iled_blinky_XiP_OTFAD.bin` - same result (not running).

Added in-memory diagnostics in `uTasker/Driver.c` function `fnDebugMsg`.
Showed test memory stick (old, slow) was timing out with "* TOD" message.
See forum discussion here:
https://community.nxp.com/t5/i-MX-RT/Trouble-configuring-uTasker-bootloader-for-RT1024/m-p/1817702
Note: Updated "TOD" messages to say "`Timeout!`" to avoid confusing Dave.
Note: Updated "No application" messages in `Applications/uTaskerSerialBoot/serial_loader.c` to avoid confusing Dave: `fnDebugMsg("No Application found in flash!!\r\n"); // DRN: improve message`

Boot-loader goes into infinite loop using fast USB stick (SanDisk ExtremePro 128GB) purchased for data-logging...

Bug Fixes per Mark Butcher:

To avoid time-out with older slow USB stick, in `Applications/uTaskerSerialBoot/usb_host_loader.c`, increase timeout value: `#define MAX_USB_MSD_READ_WAIT (5000000) // DRN: was 500000`

With the above fix slow stick works, but fast stick still goes into an infinite loop.

...