Please note, following information is for advanced users only. If you had read this info and didn't understand what is this all about, then just don't use it.

Standalone DCC Loaders do not contain any hardware initialization routines, what means it is assumed that target hardware is already initialized (DRAM/SRAM/DDR/Whatever RAM is configured and functional, FLASH memory GPIO access pins (if any) are configured, etc., etc., etc.) prior a DCC Loader is being uploaded and executed.

For example, you have a device based on the Qualcomm MSM6280 chipset; device uses NAND memory which is visible to MCU through the chipset's embedded NAND controller. Generally, upon reset, DDR memory is not visible to the core, and chipset's DDR controller has to be configured first in order to be able to access DDR RAM memory.

If device's firmware code responsible for hardware initialization is functional then it is possible to do HALT while h/w is already initialized and then upload DCC Loader. Otherwise prior the DCC Loader upload need to execute h/w initialization scripts.

Check Table1 for setup details required for each DCC Loader.

Table 1

DCC Loader File Name	RAM Base and Required Range	NAND Interface
MSM62XX_01000000_NAND.enc	0x01000000, 0x01000000...0x010FFFFF	MSM62xx (except MSM6250x)
MSM6250X_01000000_NAND.enc	0x01000000, 0x01000000...0x010FFFFF	MSM6250x
MSM72XX_01000000_NAND.enc	0x01000000, 0x01000000...0x010FFFFF	MSM72xx, Snapdragon
PXA312_5C000000_NAND.enc	0x5C000000, 0x83C00000...0x83CFFFFF	PXA312
CORTEXA8_80000000_ONENAND.enc	0x80000000, 0x80000000...0x800FFFFF	Direct OneNAND
CORTEXA8_42480000_ONENAND.enc	0x42480000, 0x42480000...0x4257FFFF	Direct OneNAND
QSC6240_01000000_NAND.enc	0x01000000, 0x01000000...0x010FFFFF	QSC6240 (and familiar)

Please note, 'XXXXXXXX's in the XXXXXXXX_RAMBASE_OneNAND.enc DCC Loader file name shows watchdog handling scheme. Here are few examples:

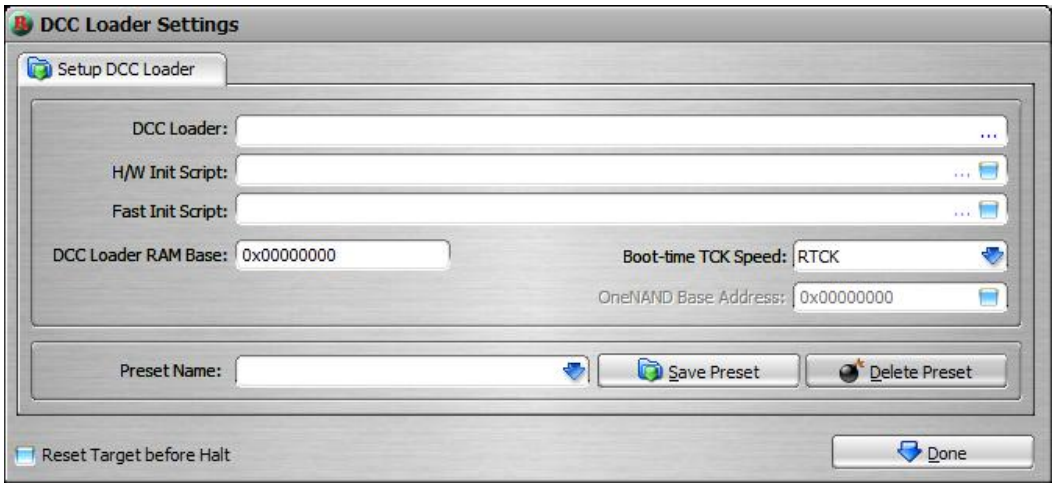
a) MSM6280_01000000_OneNAND.enc – DCC Loader code serves watchdog using MSM6280 chipset's registers and watchdog "kicking" sequence;
b) MSM6250_01000000_OneNAND.enc – DCC Loader code serves watchdog using MSM6250 chipset's registers and watchdog "kicking" sequence;
c) ARMX_01000000_OneNAND.enc – DCC Loader code does not serve watchdog. Thus it can work inside any specified ARMX target (ARM9, ARM11, CORTEXA8, etc.) which does not have watchdog or has a disabled watchdog.

For the OneNAND DCC Loader an OneNAND interface base address is additionally configured in the *DCC Loader Settings* dialog of the RIFF BOX JTAG Manager software.

So, in order to use a standalone DCC loader following steps must be performed:

1. *Custom Target Settings* must be selected;
2. In case no CMM script will be used for h/w initialization the Target, I/O Voltage and TAP# parameters have to be selected manually; in case a CMM script is used these parameters can be either set manually or to be configured by a CMM's instructions;
3. Go to DCC Read/Write page;
4. Click DCC Loader Settings button and perform required setup;
5. Use DCC Read/Write page features to access device's non volatile memory.

Below is DCC Loader Settings screenshot:



1. DCC Loader field – selects required standalone DCC Loader file;
2. H/W Init Script – selects required (if any) H/W Initialization script (can be used CMM or HAS scripts);
3. Fast Init Script – rarely used, in most cases is empty; accepts only HAS scripts;
4. DCC Loader RAM Base – RAM base where to the selected DCC Loader code will be uploaded and executed: take this address from table 1;
5. Boot-time TCK Speed – frequency used until execution is passed to the uploaded DCC Loader: when h/w is not initialized this frequency may be required to be set to much lower values than it is possible to use after the full h/w initialization is complete;
6. OneNAND Base Address – is used for OneNAND DCC Loaders; if selected OneNAND DCC Loader then this field has to be enabled and a valid OneNAND Base Address is to be entered here;
7. Reset Target before Halt – if checked will apply NRST signal before performing HALT operation. If there is no proper h/w init scripts available then this option shall not be used (unchecked).
8. Preset Name – you can store settings into preset file in order for fast and convenient switch between setups for different devices/targets;

What to do if your device has no RAM exactly at required address? For example DCC Loader's RAM base is at 0x01000000 but target has RAM mapped somewhere at other addresses? There is such option:

- Configure core's MMU module (which is available in ARM architectures starting from ARMv4 and higher) in that way, that core can access virtual memory at required base address (for this make CMM script which will create and upload translation table(s) into physical RAM, and use script's coprocessor CP15 MMU configuration instructions for MMU setup);