HPS_Software _RM

История изменений документа

Версия	Дата	Изменения
1.0	18.06.13	Начальная версия, только загрузчик
2.0	30.12.13	Основной документ

Оглавление

Автор: Степанов Д.А.

Дата создания: 18.06.2013
Состояние: актуально

1. Назначение

Документ описывает работу с системным ПО.

2. Установка инструментов

2.1.Установка SOC EDS

Необходимо скачать с https://www.altera.com/download/software/soc-eds/

установочный файл SoCEDSSetup-13.1.0.162.run и запустить его от имени пользователя на ПК под Линуксом. Ставить в home. Будет предлагать перезапустить установку скрипта от root - игнорировать предложения и продолжать. Сразу удалить папки DS5 и DS5installer так нет необходимости в нем.

В папке ~/altera/13.1/host_tools окажутся необходимые инструметы, в том числе baremetal компилятор от Mentor на базе gcc, с помощью которого собирается preloader и uboot   Исходные файлы uboot идут в виде архива который будет распакован make файлом при сборке. Неоходимые файлы для preloader генерируются по результатам сборки аппаратного проекта и включаются в дерево кода uboot.

2.2.Установка и работа с программатором USB-Blaster

После установки SOC EDS программатор оказывается в папке

/home/dmitry/altera/13.1/qprogrammer/bin

Нужно добавить конфигурацию udev в файл usb-blaster.rules.

cat   /etc/udev/rules.d/usb-blaster.rules

ACTION!="add", SUBSYSTEM!=="usb_device", GOTO="kcontrol_rules_end"

Give owner and group "rw" (MODE=66) access. Give all users "r" (MODE=4) access.

ATTRS{idProduct}=="6001", ATTRS{idVendor}=="09fb", MODE="664", GROUP="plugdev"

ATTRS{idProduct}=="6810", ATTRS{idVendor}=="09fb", MODE="664", GROUP="plugdev"

LABEL="kcontrol_rules_end"

если все нормально то бластер будет увиден, а HPS нет

1) USB-Blaster [1-5]

  Unable to read device chain (JTAG chain broken)

включаем плату появится

1) USB-Blaster [1-5]

4BA00477 SOCVHPS

02D020DD 5CSEBA6(.|ES)/5CSEMA6/..

Из под Линукса работает очень медленно, очевидно частота бластера по умолчанию 6 Мгц, увеличить нельзя.

Рекомендуется подключиться к servery под Windows там по умолчанию будет 16 Мгц. Для этого использовать утилиту jtagconfig: jtagconfig --addserver <server> <password>

С помощью программатора можно работать только с QSPI cs0, остальные недоступны.

Созданы скрипты для программирования:

preloader (prg.sh проверка ver.sh)

uboot (prg-u.sh проверка ver-u.sh)

можно писать (P), проверять (V), читать (X) по заданному адресу (-a) и задавать длинну (-s).

3. Сборка Preloader + uboot

3.1.Исходные условия для начала сборки

Должен быть собран аппаратный проект, в ходе сборке которого генерируются Hardware handoff files. Предполагается что они будут расположены в каталоге hps_isw_handoff/SOC_system_hps_0.

3.2.Описание процедуры сборки

Запустить от пользователя скрипт ~/altera/13.1./embedded/embedded_command_shell.sh

которые установит необходимые переменные окружения.

Запустить bsp-editor откроется окно GUI

Выбрать создание нового bsp, file → New

Указать путь к файлам preloader hps_isw_handoff/SOC_system_hps_0.

Поменять параметры, если необходимо и нажать generate. Нужно правильно

указать источник загрузки следующего за preloader'ом образ uboot.

Должна быть только одна галочка QSPI.

В каталоге spl_bsp будет создан makefile и каталог с генерированными файлами generated/

Зайти в каталог spl_bsp и запустить make. Будет собран образ preloader

Записать образр reloader-mkpimage.bin при помощи dd во второй раздел на sd карте, (например sdc1) или запрограммировать при помощи программатора в qspi используя скрипт prg.sh

Если в каталоге spl_bsp находится патч, то он будетавтоматически приложен после распаковки архива с исходным кодом uboot.

При повторной сборке исходный код не стирается и патч не прикладывается. Запуск bsp-editor стирает исходный код.

Рекомендуется стирать исходный код, а необходимые изменения в нем оформлять в виде патча.

Для сборки uboot выполнить комманду make uboot. Файл для прошивки называется uboot-socfpga/uboot.img (не путать с uboot.bin). Надо поменять его разрешение на bin иначе не работает программатор.

3.3.Когда надо пересобирать Preloader

Если в результате изменения в аппаратном проекте изменилось содержание файлов в папке generated, в этом можно убедится путем сравнения каталов, чтобы изменения вступили в силу необходимо пересобрать Preloader.

Можно например включить расширенную диагностику в preloader (uboot у меня при этом виснет) заменив в файле include/common.h строку 133 на if(1)

3.4.Создание патча

Можно вносить измения в preloader без использования QSYS. Поправляя одержимое generated/ и uboot-socfpga/. При этом не запускать bsp-editor, так как он затрет изменения. Окончательно оформить изменения в виде патча.

Написан скрипт patch_mk.sh который создает патч.

Кладем рядом в один каталог spl два каталога, оригинальный архивный uboot_socfpga.orig и измененный uboot_socfpga

На входе первый параметр это путь оригинальному а второй к измененому каталогу.

3.5.Текущий патч под THEA 2.2

Содержит 4 изменения:

- PHY ID для нового PHY
- замена EMAC1 на EMAC0
- добавлены сетевые параметры по умолчанию
- флаг компилятора для устранения ошибки unaligned access в uboot при попытке записи во флеш.

3.6.Ссылки на внешние документы

http://www.rocketboards.org/foswiki/Documentation/PreloaderUbootCustomization

4. QSPI

4.1.Настройка partions

На плате Thea 2.2 может быть установлено до 4 микросхем n25q00 по 128 МБ каждая. Адреса разделов настраиваются в описание binary blob для ядра (thea_22.dts)

Выделяется на устройстве 0 первый раздел под RAW DATA без файловой системы размером 8M. Остальные впоследствии могут форматироваться под JFFS2.

4.2.Карта памяти partion 0

0x000000 — 0x03FFFF preloader 4 копии

0x040000 — 0x05FFFF сохраняемое окружение uboot

0x060000 — 0x0EFFFF uboot

0x0F0000 — 0x0FFFFF fdt-image (binary blob)

0x100000 — 0x7FFFFF Linux kernel

4.3.Запись данных QSPI

RAW DATA Partion 0 записывается при помощи программатора quartus_hps в соответствии с картой памяти выше. При наличии уже записанного программатором uboot можно писать через него используя комманды sf. (смортри sf help)

sf probe 0:0 (0:1 0:2 0:3) — выбирает микросхему

sf erase

sf write

На уже отформатированные разделы запись осуществляется из Linux после монтирования раздела

4.4.Работа из под Linux

На target машине должен быть установлен пакет mtd-utils.

Форматировать раздел под JFFS2 коммандой :

flash_erase -j /dev/mdt<x> 0 0

затем раздел нужно примонтировать:

mount -t jffs2 /dev/mtdblock<x> /mnt/f<x>p<y>

Есть еще один способ (не проверялось).

Создается файл содержащий внутри себя файловую систему jffs2 при помощи утилиты mkfs.jffs2 из пакета mtd-utils.

Записываем этот файл на нужную partion (не обязательно форматированную)

flashcp image.jffs2 /dev/mtd<x>

5. FDT – binary blob

Механизм нужный для сообщения стандартным драйверам ядра особенностей конкретной платы (наличие и адреса устройств).

Адрес blob'a в памяти передается ядру (параметр после дефиса) при загрузки из uboot.

Например bootz 0x7fc0 — 0x100

Altera предлагает автоматизированный путь

- утилитой sopc2fdt из файлов описания платы, описания clock, описания проекта (.sopcinfo) строится текстовое представление .dts
- при помощи компилятора dtc создается бинарное представление dtb

Автоматизированный путь не удалось осуществить, не работает утилита sopc2 fdt повидимому неверный синтаксис описания платы. Синтаксис зависит от версии ядра.

Удачный путь взяли бинарный блоб от GSRD и от EBV откомпилировал в текстовый вид с помощью dtc, внес исправления, затем обратная компиляция.

Можно сделать один раз для новой платы, для своих устройств вносить изменения не надо. Даже описывать bridge не обязательно, так будет испрользоваться свой собственный драйвер.

Текущий файл с текстовым представлением thea_22.dts

6. Linux

6.1.NFS

Удобно для отладки. Запущен сервер на машине sda 192.168.193.55

каталог для монтирования ~/dmitry/mnt/socmount задается в скриптах uboot nfsboot. Ядро и блоб загружаются по tftp с сервера australia, есть скрипт uboot nfsload.

Сейчас установлен образ на основе Debian. Грузится по умолчанию при старте системы

Планируется перенести на сервер Australia.

6.2.QSPI файловая система JFFS2

Образ Linux (63 M) от демо платы Socrates (EBV) установлен на partion 1.

Для загрузки остановить автозапуск в uboot и выполнить скрипты

run qspiload

run qspiboot

в настройках хоста надо разрешить подключение с пустым паролем SSH.

6.3.QSPI образ для RAM disk

Предполагается как основной способ работы. Создаем образ RAM диска с файловаой системой ext2 и записываем ее в QSPI при помощи uboot.

При старте загружаем в память и указываем ядру root=/dev/ram

монтируем остальные partions потом чтобы иметь энергонезависимую память

7. Собственные устройства

Собственные устройства подключение через lw-hps-2-fpga bridge. Имеют базовое смещение 0xFF200000. Не требуется перекомпиляция preloader и измениния FDT blob. Надо писать свой драйвер в kernel space.

Сейчас реализовано устройство по адресу 0, простой счетчик обращений.

8. Прерывания

GIC — General Interrupt Controller описан в документах на ARM.

По-видимому нельзя управлять полярностью сигнала запроса на прерывания, только выбрать фронт или уровень. Смотреть в ядре еще раз.

От FPGA идут 64 линии разбитых на 2 слова irq0 irq1.

Номера прерываний с 72 по 136. На схеме отображаются номерм на 32 меньше, так как первые 32 прерывания под собственные нужды. В блобе номера тоже на 32 меньше. Irq0[0] — соответствует номеру 72 и так далее.

Для отладки можно смотреть в регистрах GIC непостредственно:

0xFFFED108 — 0x100 (8 бит для 72 прерывания) разрешение прерывания (вызывает срабатывание прерывания если оно pending)

можно посмотреть cat /proc/interrups

0xFFFED208 — 0x100 (8 бит для 72 прерывания) set pending (запись 1 устанавливает pending)

0xFFFED288 — 0x100 (8 бит для 72 прерывания) clear pending (запись 1 сбрасывает pending)

0xFFFEDC10 — бит 16,17 (или 144,145 биты от начала, по два бита на прерывания) управление фронт\уровень

полярность -???

Получилось сделать пока только так, чтобы исходное значение от аппаратного устройства все 1, то есть все прерывания pending, нужное можно сбросить и потом разрешить, иначе не работает. Разобраться при написании драйвера.

Бит 0 в регистре 0xFF20000C (наше собственное устройство) установка в 1 вызывает прерывание.

9. ECC

Собрали аппаратный проект с его поддержкой и preloader.

Необходимо осуществить первоначальное обнуление SDRAM.

В uboot по-видимому работает. Необходимо дальнейшее тестирования.

Есть регистры из которых можно извлечь статистику срабатываний.

10. Ядро

На сайте http://www.rocketboards.org/foswiki/Documentation/GsrdGitTrees

есть репозиторий Git с исходным кодом. Понадобится для сборки своих драйверов.

Копируем себе репозиторий

$ git clone http://git.rocketboards.org/linux-socfpga.git

Вычекиваем по тагу релиза 13.1 GSRD

```
$ cd linux-socfpga
$ git checkout ACDS13.1_REL_GSRD_PR

Собираем ядро, должен быть установлен компилятор от Linaro

make CROSS_COMPILE=arm-linux-gnueabihf- ARCH=arm mrproper
make CROSS_COMPILE=arm-linux-gnueabihf- ARCH=arm socfpga_defconfig
make CROSS_COMPILE=arm-linux-gnueabihf- ARCH=arm zImage
```

11. uboot

На сайте http://www.rocketboards.org/foswiki/Documentation/GsrdGitTrees

есть репозиторий Git с исходным кодом. Понадобится для подключения

HPS_Software _RM

изменений.

12. Загрузка FPGA через FPGA master

Есть возможность загружать FPGA при помощи HPS через FPGA master.

Загружать можно из preloader или из uboot. Проверено из uboot.

Длину образа при загрузке не указывать, определяется автоматически. После загрузки необходимо разрешить работу моста lwHPS2FPGA, для этого в регистре L3remap (0xff800000) разрешить видимость записав 0x11, и снять его со сброса в регистре (0xffd0501c) записав 0x5.

В uboot сделан скрипт для загрузки fpga:

fpgaload=tftp ${fpgadata} ${tftpdir}${fpgaimage}; fpga load 0 ${fpgadata}; run bridge_enable_handoff;

13. Установка компилятора для сборки ядра и приложений

Берем исполняемые файлы с launchpad.net

gcc-linaro-arm-linux-gnueabihf-4.8-2013.10_linux.tar.bz2

разворачиваем архив в каталог ~/bin

прописываем PATH в файле .profile у пользователя

```
if [ -d "$HOME/bin" ] ; then
    PATH="$HOME/bin:$HOME/bin/eclipse-4.3-cpp:$HOME/bin/gcc-linaro-arm-linux-gnueabihf-4.8-2013.10_linux/bin:$PATH"
fi
```

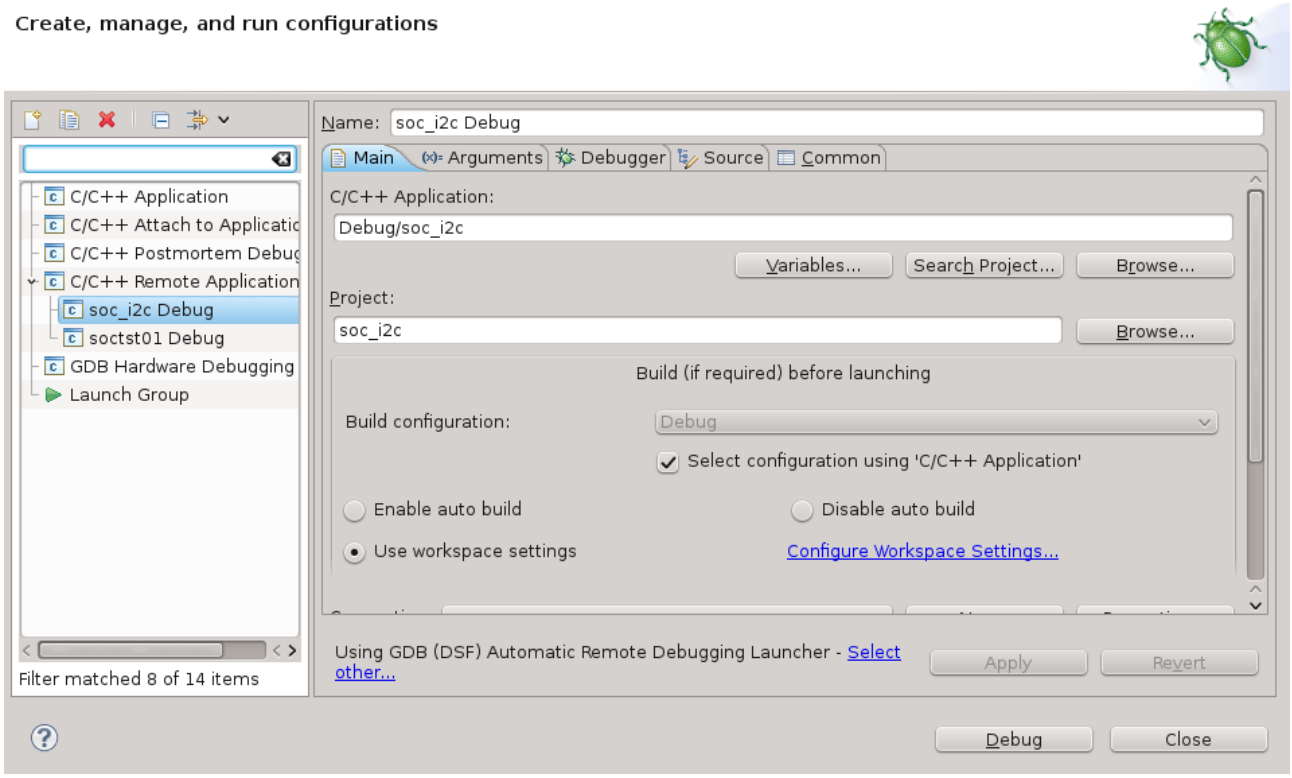
14. Проект в Eclipse 4.3

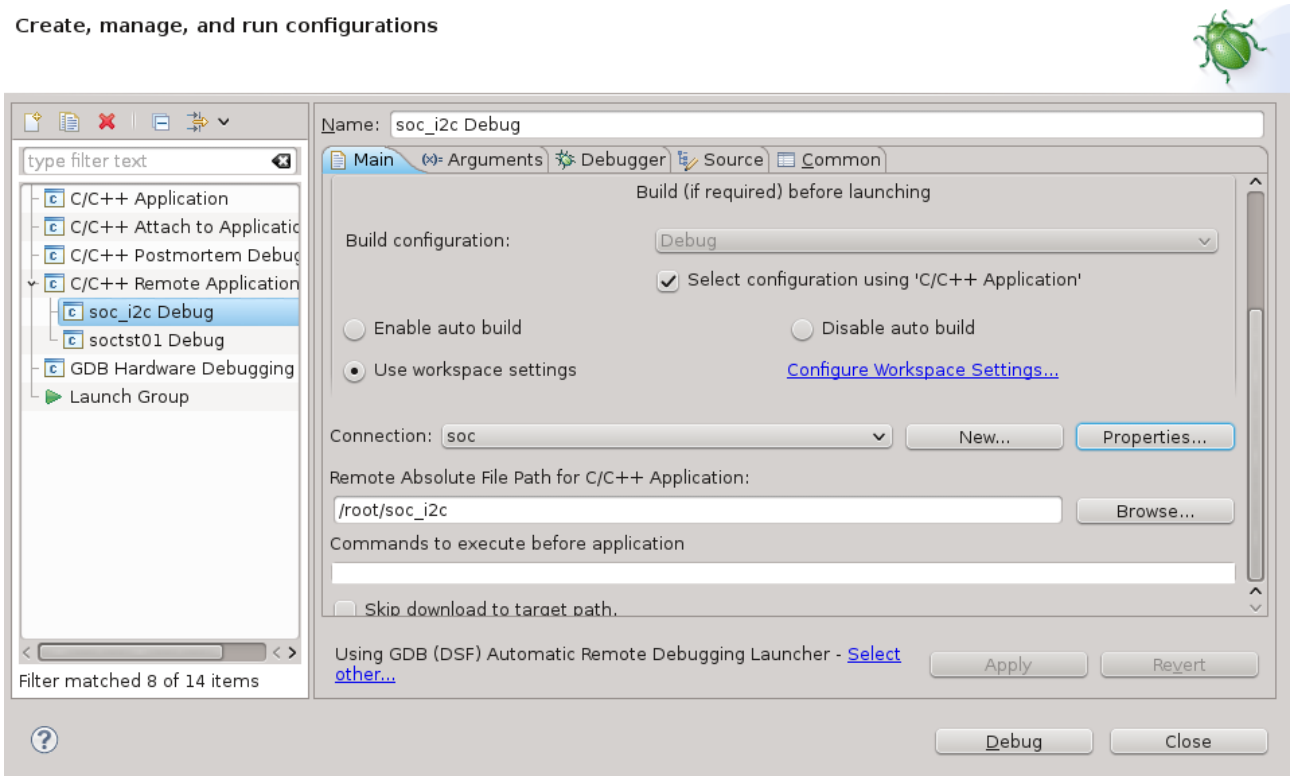
В Eclipse установлены плагины CROSS-COMPILE и REMOTE APPLICATION

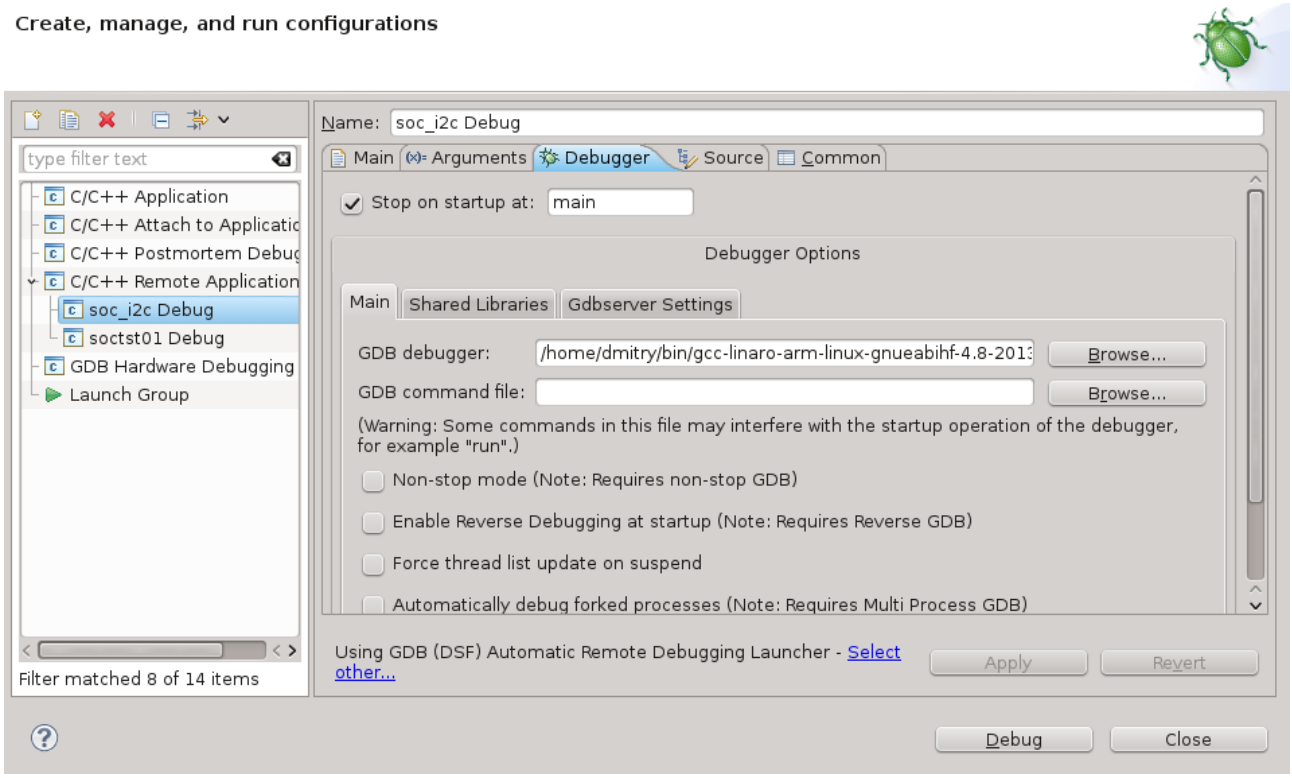
При создании проекта указываем на cross tools.

Для запуска и отладки на удаленной машине настраивается новый C/C++ Remote Application debug configuration.

В нем надо настроить


Create, manage, and run configurations
Name: soc_i2c Debug
Main
Arguments
Debugger
Source
Common
C/C++ Application
C/C++ Attach to Applicatic
C/C++ Postmortem Debug
C/C++ Remote Application
soc_i2c Debug
soctst01 Debug
GDB Hardware Debugging
Launch Group
C/C++ Application:
Debug/soc_i2c
Variables...
Search Project...
Browse...
Project:
soc_i2c
Browse...
Build (if required) before launching
Build configuration:
Debug
Select configuration using 'C/C++ Application'
Enable auto build
Disable auto build
Use workspace settings
Configure Workspace Settings...
Using GDB (DSF) Automatic Remote Debugging Launcher - Select other...
Apply
Revert
Filter matched 8 of 14 items
Debug
Close


Create, manage, and run configurations
Name: soc_i2c Debug
type filter text
Main
Arguments
Debugger
Source
Common
C/C++ Application
C/C++ Attach to Applicatic
C/C++ Postmortem Debug
C/C++ Remote Application
soc_i2c Debug
soctst01 Debug
GDB Hardware Debugging
Launch Group
Build (if required) before launching
Build configuration:
Debug
Select configuration using 'C/C++ Application'
Enable auto build
Disable auto build
Use workspace settings
Configure Workspace Settings...
Connection:
soc
New...
Properties...
Remote Absolute File Path for C/C++ Application:
/root/soc_i2c
Browse...
Commands to execute before application
Skip download to target path.
Using GDB (DSF) Automatic Remote Debugging Launcher - Select other...
Apply
Revert
Filter matched 8 of 14 items
Debug
Close


Create, manage, and run configurations
type filter text
C/C++ Application
C/C++ Attach to Applicatic
C/C++ Postmortem Debug
C/C++ Remote Application
soc_i2c Debug
soctst01 Debug
GDB Hardware Debugging
Launch Group
Filter matched 8 of 14 items
Name: soc_i2c Debug
Main
Arguments
Debugger
Source
Common
Stop on startup at: main
Debugger Options
Main
Shared Libraries
Gdbserver Settings
GDB debugger: /home/dmitry/bin/gcc-linaro-arm-linux-gnueabihf-4.8-201
Browse...
GDB command file:
Browse...
(Warning: Some commands in this file may interfere with the startup operation of the debugger, for example "run".)
Non-stop mode (Note: Requires non-stop GDB)
Enable Reverse Debugging at startup (Note: Requires Reverse GDB)
Force thread list update on suspend
Automatically debug forked processes (Note: Requires Multi Process GDB)
Using GDB (DSF) Automatic Remote Debugging Launcher - Select other...
Apply
Revert
Debug
Close