The following document contains information on Cypress products. The document has the series name, product name, and ordering part numbering with the prefix “MB”. However, Cypress will offer these products to new and existing customers with the series name, product name, and ordering part number with the prefix “CY”.

How to Check the Ordering Part Number

1. Go to www.cypress.com/pcn.
2. Enter the keyword (for example, ordering part number) in the **SEARCH PCNS** field and click **Apply**.
3. Click the corresponding title from the search results.
4. Download the Affected Parts List file, which has details of all changes

For More Information

Please contact your local sales office for additional information about Cypress products and solutions.

About Cypress

Cypress is the leader in advanced embedded system solutions for the world's most innovative automotive, industrial, smart home appliances, consumer electronics and medical products. Cypress' microcontrollers, analog ICs, wireless and USB-based connectivity solutions and reliable, high-performance memories help engineers design differentiated products and get them to market first. Cypress is committed to providing customers with the best support and development resources on the planet enabling them to disrupt markets by creating new product categories in record time. To learn more, go to www.cypress.com.

AN205441

FCR4, MB9EF126, Debug Requirements for Tool Chain

For debugging the Arm® CoreSight® interface is used in Arm Cortex® R4 MCUs. Some additional features are added to enhance the debug possibilities and security. Following chapter will describe these features and the requirements for the debug tool chain in order to support these features

Contents

1 Introduction

For debugging the ARM CoreSight® interface is used in ARM Cortex R4 MCUs. Some additional features are added to enhance the debug possibilities and security. Following chapter will describe these features and the requirements for the debug tool chain in order to support these features.

2 Debug Features and Requirements

This chapter describes the additional debug features and requirements of FCR4 devices

2.1 CoreSight AHB access port

Besides the non-invasive trace functionality an AHB access port (AHB-AP) of the CoreSight system is connected to the bus fabric and allows for non-invasive debugging – or minimal-invasive since the AHB-AP bus master has lowest master priority.

For peripheral register access it is recommended to use the AHB access port. Some peripherals may evaluate the master ID of the access and e.g. will not change FIFO read pointers or clear flags on reading.

The debug tool chain should provide a way to select how the accesses (i.e. using AHB-AP or ARM core) are performed.

2.2 CoreSight APB access port

The CoreSight in Calypso also offers an APB access port for accessing the dedicated debug bus. The base addresses of the debug components can be found in the memory map or by evaluating the DAP ROM table. The SecurityChecker address space that is important for the debugger connection sequence starts at address 0xF000.

The debug tool chain should provide a way to access the debug bus, e.g. if data shall be exchanged between user and application on a secured device via SCCFG_GPREG0~1.

2.3 General overview about BootROM flow

After any reset the BootROM code is executed. Access to the device via debug interface is completely blocked (with the exceptions mentioned further below) until the device security settings have been evaluated. After this evaluation the blocking may

- persist forever
- persist until the user application allows security key entering and a security key has been entered successfully
- persist until a security key has been entered successfully
- be removed depending on the configured security markers in the Flash memory.

The following resources are accessible at any time with the debugger by using the APB access port (APB-AP) of CoreSight:

- DAP ROM table
- SCCFG_STAT2 (Security Checker Status register 2)
- SCCFG_SECKEY0~3 (Security Checker Security Key register 0~3)
- SCCFG_GPREG0~1 (Security Checker General Purpose register 0~1)
- SCCFG_UNLCK (Security Checker Unlock register), writing the correct unlock key to this register is required before write accesses to other Security Checker registers are accepted

The point in time when BootROM has finished the evaluation is indicated by the status bit SCCFG_STAT2_DBGRDY.

Any debugger access to other resources than mentioned above must wait until SCCFG_STAT2_DBGRDY is ‘1’ and device is either unsecured or secured but security key has been entered successfully. Note, this also means that security key can be entered at any point of time, e.g. while BootROM is still evaluating security settings.

At the end of the BootROM code an optional waiting time may be enabled with a certain marker in the Flash memory. Afterwards the BootROM will branch to the user application.

2.4 BootROM waiting time for debugger

There is a Boot Description Record (BDR) in the Flash memory. Besides other entries this BDR also contains a so-called marker that determines if the BootROM will wait a certain time before branching to the user application.

By programming a “magic word” to this marker in the Flash memory the waiting time is skipped. If the marker value does not match this magic word – e.g. if Flash is empty – the waiting time is applied. Before it is applied it is checked if a debugger is connected at all otherwise the waiting time is also skipped.

Conditions that leave or skip the waiting loop are:

- Magic word in Flash marker is set.
- No debugger connection was recognized. The MCU assumes that a debugger is connected if at least 16 JTAG clock cycles at the TCLK pin have been recognized (status bit: SYSC_JTAGDETECT_DBGCON)
- Debugger sets bit SYSC_JTAGCNFG_DBGDONE to indicate that the debug configuration has finished (e.g. trace, breakpoint settings)
- Half of the default watchdog interval has elapsed (since reset). This ensures that for the user application at least the other half of the default watchdog interval is remaining for watchdog configuration

The minimum default watchdog interval after reset is dependent on the maximum possible RC clock frequency (nominal frequency + “plus-tolerance”).

If no other conditions mentioned above apply, the BootROM will branch to the user application after a certain amount of time. In this time frame the debugger must also complete all necessary configurations:

$$\text{Maximum configuration time after reset} = \frac{1}{2} * \frac{\text{Initial watchdog counter value}}{\text{Maximum RC frequency}}$$

Initial watchdog counter value: 0x01000000

Maximum RC frequency = Nominal frequency + Plus-Tolerance

= 12 MHz + 100% * 12 MHz =24 MHz

Maximum configuration time after reset = 350 ms

Most debugger accesses are blocked until a certain point in the BootROM execution flow that is indicated by the SCCFG_STAT2_DBGRDY status bit (refer chapter 0). The BootROM execution time up to this point must be subtracted from the calculated 350ms.

The BootROM execution duration up to this point is not specified but since it is very small compared to the 350ms calculated above it can be neglected for most cases.

2.5 Establishing debug connection to MCU with empty Flash

In case a connection to a MCU with empty Flash memory shall be done, debugger needs to ensure that the ARM core is halted after the connection has been successfully established.

During the waiting time explained in chapter 2.4 the debugger must connect, disable the watchdog and halt the ARM core. After that e.g. the Flash programming kernel can be downloaded.

If the core is not halted in time the BootROM will branch to an “empty” Flash address and therefore fetches 0xFFFFFFFF as instruction code. Since 0xFFFFFFFF is an undefined instruction an exception handler is called. In the default exception handler implemented in the BootROM a software-triggered hardware reset will be executed.

2.6 Hardware Watchdog

2.6.1 Overview and general requirements

The FCR4 series supports an RC oscillator based Hardware Watchdog. This watchdog is enabled by default and its configuration needs to be locked before the default watchdog interval has elapsed, otherwise the MCU is reset.

The configuration options allow the user to completely disable the watchdog or to enable a debug mode. If the debug mode is enabled the watchdog timer will be halted while the core is in debug state.

After every reset the configuration can only be adjusted and locked once. After it has been locked no further writes are possible to the configuration registers and they will result in an error response.

All the points mentioned above must be regarded in the following situations:

- Flash is empty: Since no user application is configuring/clearing the watchdog, it should be disabled and the configuration must be locked afterwards. Then, e.g. the Flash programming kernel can be downloaded
- User application is programmed that configures the watchdog at a certain point in the application start-up code. User should enable debug mode during application development.

- Debugger should not halt the core before the application has locked the watchdog configuration.
- If it is necessary to debug the application start-up code up to the point where the watchdog is configured, debugger must configure the watchdog appropriately. If the application is not aware of this problem and gets to the point where it tries to write the locked watchdog registers a data abort exception will be raised. (Workaround: application may check if watchdog registers are already locked before it tries to write the configuration registers)

  For above stated reasons the debugger user needs to have the possibility to set the required watchdog configuration via debugger. In case of script file support of a tool chain this could probably be done with a script.

2.6.2 Stopping Watchdog during breakpoint

The bit WDG_CFG_DEBUGEN defines if watchdog is stopped during breakpoint or not. This bit should be set during application development. If this bit is set debugger does not need to trigger watchdog during breakpoint.

As an option debugger could check register configuration given by user and notify if this bit is not set.

2.7 Debugger connection sequence

It is possible to close the debug port for security reasons. A secured debug interface requires the debugger user to send a valid security key in order to open the debug port.

Security settings are configured with markers in the Flash memory.

It is possible to identify if the debug port is secured by reading SCCFG_STAT2 register. This register can be read at any time via the debug bus (APB access port). Following information can be obtained by reading this register:

- Bit 0 (SECEN) identifies if the device is secured (‘1’) or not (‘0’).
- Bit 1 (SECLOCK) identifies if the device is locked (‘1’) or unlocked (‘0’).
- Bit 2 (SEC) identifies if the Security key can be entered (‘1’) or not (‘0’).
- Bit 8 (DBGRDY) identifies if the device is ready for debug access (‘1’) or not (‘0’). This bit indicates the end of the BootROM security evaluation. Certain resources are also accessible while DBGRDY is ‘0’.

After reset there is a certain time while debug port blocks most other accesses. Refer chapter 0 for more information.

The user and/or debugger must be aware of following three potential situations and behave accordingly when connecting to the target MCU:

- It is known that the target MCU is unsecured
- It is known that the target MCU is secured
- The security status of the target MCU is unknown (generic approach)

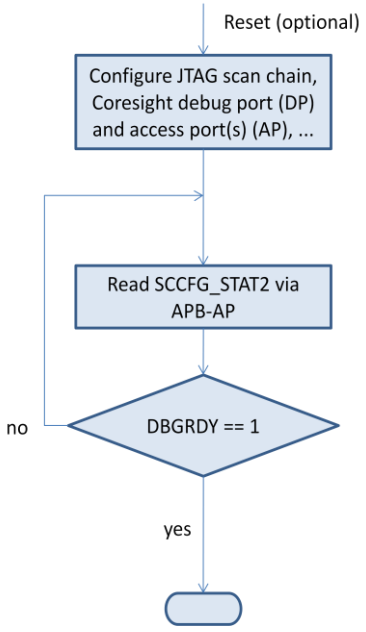
These 3 possibilities are covered in the following sub-chapters.

Furthermore, the debugger must be aware that most types of resets cause a re-evaluation of the security setting by the BootROM, and hence unlocking the device may also be necessary during a debug session if such a reset had occurred. Information about which reset types cause a re-evaluation of security can be found in sub chapter “Effect of reset sources” of System Controller chapter in the Hardware Manual.

2.7.1 MCU unsecured

Although target MCU is actually unsecured debugger must wait until BootROM has finished security evaluation before debugger is accessing any resources other than those that are always accessible.

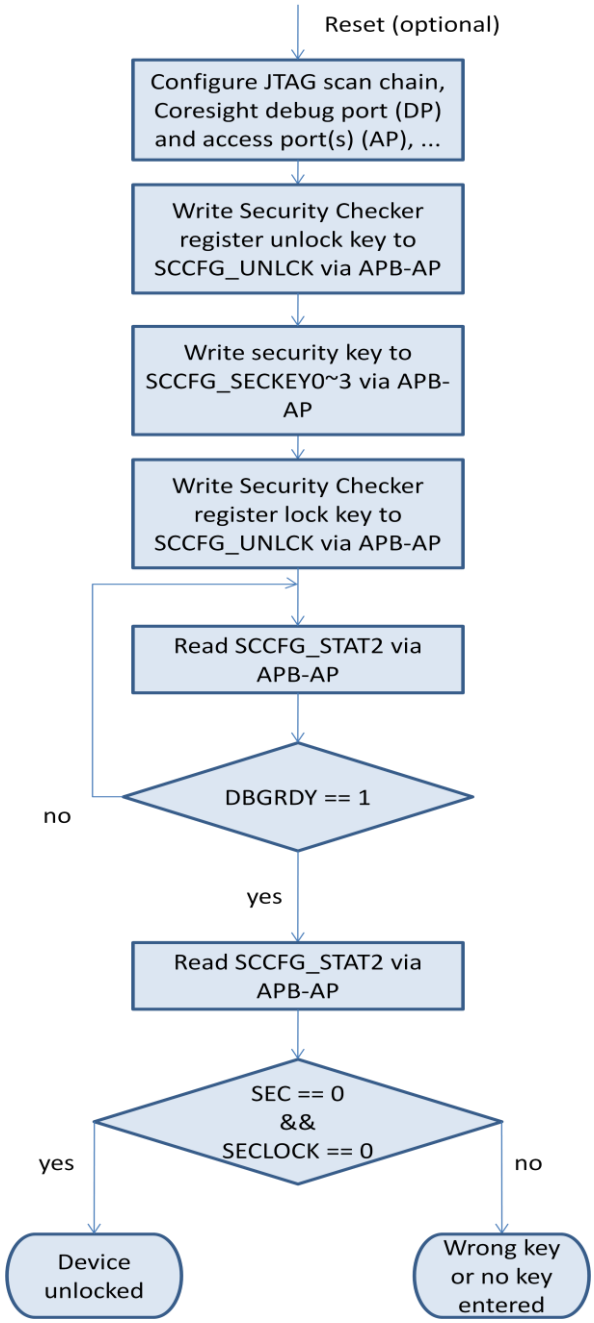
Figure 2-1. Connection sequence if MCU is unsecured



2.7.2 MCU secured

Security key can be entered at any time. The key comparison and its result can be obtained when SCCFG_STAT2_DBGRDY is '1'

Figure 2-2. Connection sequence if MCU is secured

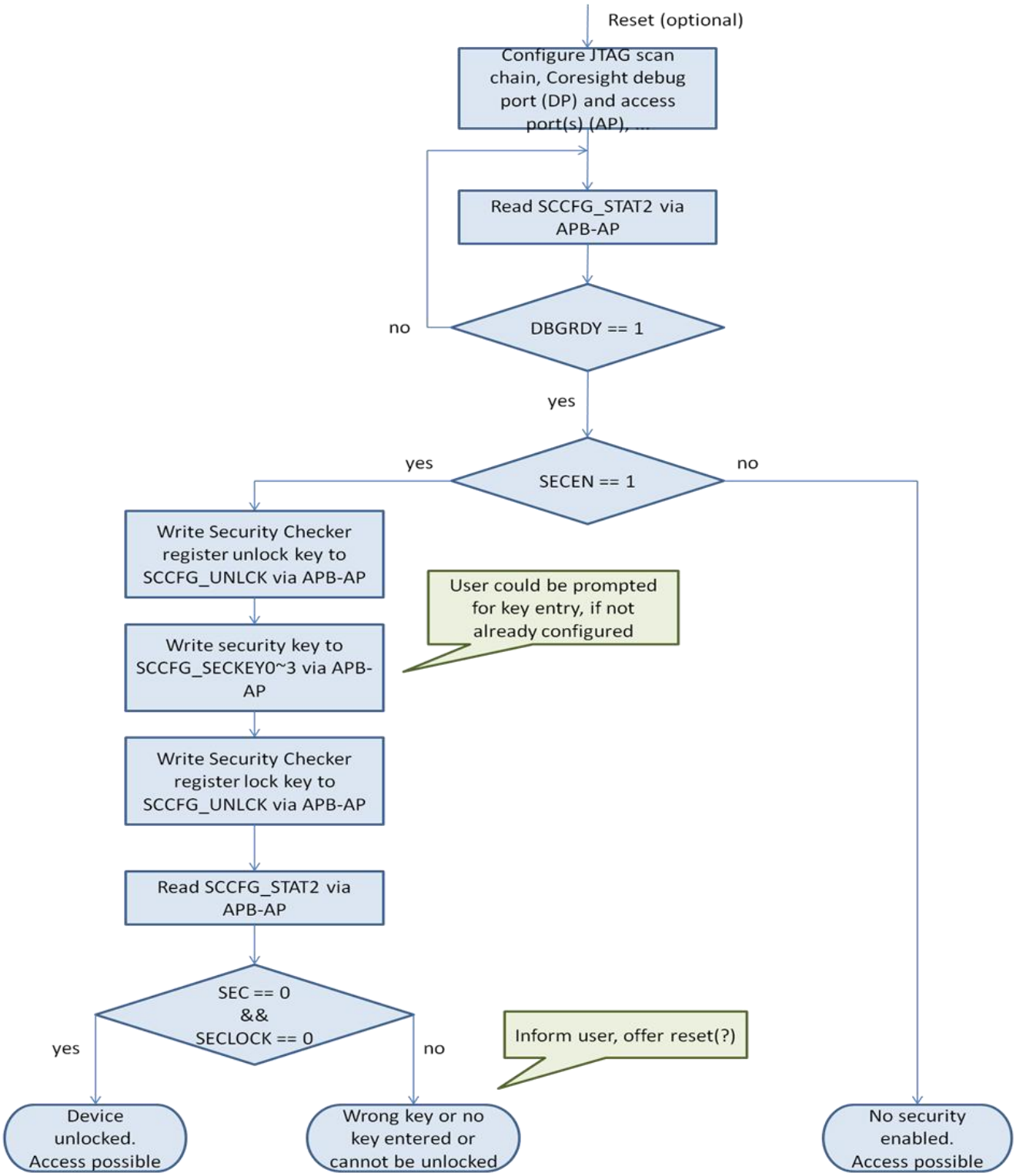


In case a wrong key has been entered no further key entry is possible without resetting the MCU

2.7.3 MCU state unknown, generic sequence

Following flow chart shows the generic connection sequence

Figure 2-3. Generic Connection Sequence



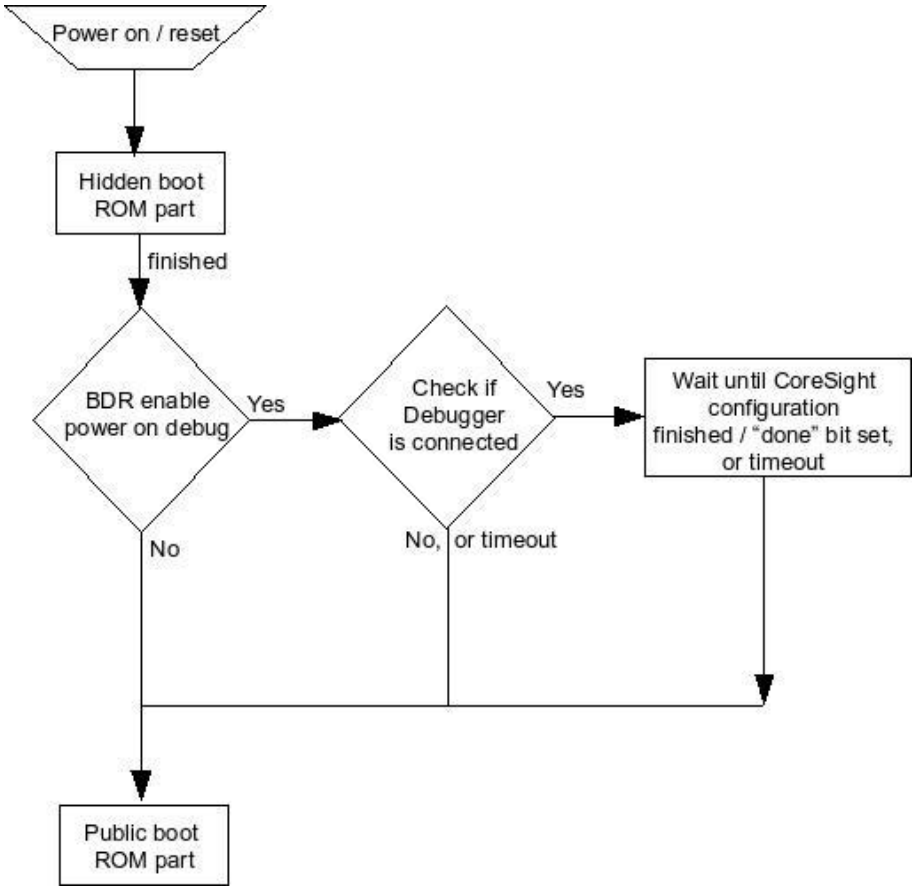
2.8 Power-on debugging

It is possible to do power-on debugging with the FCR4 series. Since the debug configuration is lost after a power on reset it is normally not possible to .e.g. setup the trace component for tracing the first user instructions. This can be achieved by using the waiting time mechanism that is described in chapter 2.4.

First time configuration of the CoreSight system can be done as follows:

After the hidden part of the BootROM is executed, the BDR is evaluated. In case the debugger waiting time is enabled and a debugger is connected, the SYSC_JTAGDETECT_DBGCON bit is set to "1". In this case the hardware initializes the SYSC_JTAGCNFG_DBGDONE bit to "0". After the debugger has finished the configuration of all necessary CoreSight components, the debugger has to set the SYSC_JTAGCNFG_DBGDONE bit to "1". Now the boot resumes with public boot ROM part. If a timeout occurs before the DBGDONE bit is set the boot process resumes also. The check if a debugger is connected is done by counting 16 TCK clocks. In case no debugger is connected or a timeout occurs, the boot process resumes. The CoreSight System uses TRST for reset, so the CoreSight configuration is still valid after system resets other than power on reset.

Figure 2-4. Power on debug flow chart (BootROM)



The MCU has several internal power-domains which can be enabled / disabled separately to have a flexible low power mode for different use cases. The debug and trace components are located in power domain PD2 and PD3. Within the System Controller there is a Fake Power Down bit. Setting this bit will keep PD2 and PD3 active even if application is disabling these power domains.

2.8.1 Required registers

The MCU includes a SYSC_JTAGDETECT register, which is used for debugger connection status

Bit 0, DBGCON bit, signals that a connection is recognized; this bit is set / cleared by MCU.

Table 2-1. DBGCON bit description

Bit value	Description
0	Debugger unconnected (initial value after reset)
1	Debugger connected

The JTAG configuration register (SYSC_JTAGCNFG) has a configuration bit to show that CoreSight configuration is completed. This register is initialized to '0', when debugger is connected. This bit is set by the debugger after the debug configuration is done. This register bit is read by BootROM to check the status of debug configuration.

The DBGDONE bit signals if CoreSight configuration is done.

Table 2-2. DBGDONE bit description

Bit value	Description
0	Debugger configuration ongoing
1	Debugger not connected or configuration done (initial value after reset)

2.9 Black list support

The tool chain needs to support a black / or white list for memory accesses.

The FCR4 MCU will apply an abort when accessing reserved areas. For some legacy peripheral memory areas (peripheral bus 0, 1 and 3) a special unit, called Bus Error Collection Unit (BECU) is responsible for issuing an NMI and collecting information about the access if a connected peripheral reports an access violation.

For this reason it is mandatory that the tool chain can apply a black list for given regions e.g. memory window, watch window to ensure that only accesses to allowed memory areas are executed by the tool chain.

Following features are required:

Table 2-3. Required Black list features

Feature	Required	Details
Minimum region size	Byte (8-bit)	Also within a peripheral there are reserved areas
Write/ Read access	Selectable	Should be possible to define also if read / write / or both should not be done
Access width	Selectable 8-bit, 16-bit, 32-bit, 64-bit and all combination	Should be possible to define which access width is not allowed to use. As some of the registers also report violations when accessing with wrong access width. In addition, multiple access width should be able to set.

3 Appendix

3.1 References

Related documents for further information are listed in the table below:

Ref. #	Document file name	Description
1	FCR4_Cluster_Series_Hardware_Manual.pdf	FCR4 Cluster Series Hardware Manual
2	MB9EF126-MN707-00001-0v??-E.pdf	FCR4 Cluster Series MB9EF126 - CALYPSO Datasheet
3	DDI0363D_cortexr4_r1p3_trm.pdf	ARM Cortex-R4 and Cortex-R4F Technical Reference Manual
4	DDI0406B_arm_architecture_reference_manual_errata_markup_4_0.pdf	ARM® Architecture Reference ManualARM®v7-A and ARM®v7-R edition
5	IHI0031A_ARM_debug_interface_v5.pdf	ARM® Debug Interface v5Architecture Specification
6	IHI0029B__CoreSight_ArchSpec.pdf	CoreSight™v1.0Architecture Specification
7	CoreSightComponentsTechnicalReferenceManual.pdf	CoreSight™ ComponentsTechnical Reference Manual

Document History

Document Title: AN205441 - FCR4, MB9EF126, Debug Requirements for Tool Chain

Document Number:002-05441

Revision	ECN	Orig. of Change	Submission Date	Description of Change
**	-	MKEA	11/25/2010	MSt, 1.0 First release
			02/14/2010	CEy, 1.1 Several changes
			05/02/2013	MSt, 1.2 Removed confidential remark
*A	5074257	MKEA	01/14/2016	Converted Spansion Application Note “MCU-AN-300125-E-V12” to Cypress format
*B	5870463	AESATMP9	09/01/2017	Updated logo and copyright.
*C	6054547	NOFL	02/12/2018	Updated links Updated template

Worldwide Sales and Design Support

Cypress maintains a worldwide network of offices, solution centers, manufacturer's representatives, and distributors. To find the office closest to you, visit us at Cypress Locations.

Products

Arm® Cortex® Microcontrollers	cypress.com/arm
Automotive	cypress.com/automotive
Clocks & Buffers	cypress.com/clocks
Interface	cypress.com/interface
Internet of Things	cypress.com/iot
Memory	cypress.com/memory
Microcontrollers	cypress.com/mcu
PSoC	cypress.com/psoc
Power Management ICs	cypress.com/pmic
Touch Sensing	cypress.com/touch
USB Controllers	cypress.com/usb
Wireless Connectivity	cypress.com/wireless

PSoC® Solutions

PSoC 1 | PSoC 3 | PSoC 4 | PSoC 5LP | PSoC 6 MCU

Cypress Developer Community

Community | Projects | Videos | Blogs | Training | Components

Technical Support

cypress.com/support

Cypress Semiconductor
198 Champion Court
San Jose, CA 95134-1709