ACPI for CoreSight™ 1.2

Platform Design Document

Non-confidential

arm

Document number: DEN0067

Contents

Release information

Date	Version	Changes
2023/May/23	1.2	• Use of inclusive language • Fixed issues with references and added missing references • Updated graph example to match ACPI _DSD guide • Added _HID definition for Embedded Trace Extension (ETE) • Clarify in Section 2.3 that MMIO interface is mandatory only when the device is not accessible via system instructions
2019/Jul/12	1.1	• External release
2019/May/20	1.0	• Added CoreSight static funnel

Arm Non-Confidential Document Licence (“Licence”)

This Licence is a legal agreement between you and Arm Limited (“**Arm**”) for the use of Arm’s intellectual property (including, without limitation, any copyright) embodied in the document accompanying this Licence (“**Document**”). Arm licenses its intellectual property in the Document to you on condition that you agree to the terms of this Licence. By using or copying the Document you indicate that you agree to be bound by the terms of this Licence.

“**Subsidiary**” means any company the majority of whose voting shares is now or hereafter owner or controlled, directly or indirectly, by you. A company shall be a Subsidiary only for the period during which such control exists.

This Document is **NON-CONFIDENTIAL** and any use by you and your Subsidiaries (“Licensee”) is subject to the terms of this Licence between you and Arm.

Subject to the terms and conditions of this Licence, Arm hereby grants to Licensee under the intellectual property in the Document owned or controlled by Arm, a non-exclusive, non-transferable, non-sub-licensable, royalty-free, worldwide licence to:

(i) use and copy the Document for the purpose of designing and having designed products that comply with the Document;

(ii) manufacture and have manufactured products which have been created under the licence granted in (i) above; and

(iii) sell, supply and distribute products which have been created under the licence granted in (i) above.

Licensee hereby agrees that the licences granted above shall not extend to any portion or function of a product that is not itself compliant with part of the Document.

Except as expressly licensed above, Licensee acquires no right, title or interest in any Arm technology or any intellectual property embodied therein.

THE DOCUMENT IS PROVIDED “AS IS”. ARM PROVIDES NO REPRESENTATIONS AND NO WARRANTIES, EXPRESS, IMPLIED OR STATUTORY, INCLUDING, WITHOUT LIMITATION, THE IMPLIED WARRANTIES OF MERCHANTABILITY, SATISFACTORY QUALITY, NON-INFRINGEMENT OR FITNESS FOR A PARTICULAR PURPOSE WITH RESPECT TO THE DOCUMENT. Arm may make changes to the Document at any time and without notice. For the avoidance of doubt, Arm makes no representation with respect to, and has undertaken no analysis to identify or understand the scope and content of, third party patents, copyrights, trade secrets, or other rights.

NOTWITHSTANING ANYTHING TO THE CONTRARY CONTAINED IN THIS LICENCE, TO THE FULLEST EXTENT PETMITTED BY LAW, IN NO EVENT WILL ARM BE LIABLE FOR ANY DAMAGES, IN CONTRACT, TORT OR OTHERWISE, IN CONNECTION WITH THE SUBJECT MATTER OF THIS LICENCE (INCLUDING WITHOUT LIMITATION) (I) LICENSEE’S USE OF THE DOCUMENT; AND (II) THE IMPLEMENTATION OF THE DOCUMENT IN ANY PRODUCT CREATED BY LICENSEE UNDER THIS LICENCE). THE EXISTENCE OF MORE THAN ONE CLAIM OR SUIT WILL NOT ENLARGE OR EXTEND THE LIMIT. LICENSEE RELEASES ARM FROM ALL OBLIGATIONS, LIABILITY, CLAIMS OR DEMANDS IN EXCESS OF THIS LIMITATION.

This Licence shall remain in force until terminated by Licensee or by Arm. Without prejudice to any of its other rights, if Licensee is in breach of any of the terms and conditions of this Licence then Arm may terminate this Licence immediately upon giving written notice to Licensee. Licensee may terminate this Licence at any time. Upon termination of this Licence by Licensee or by Arm, Licensee shall stop using the Document and destroy all copies of the Document in its possession. Upon termination of this Licence, all terms shall survive except for the licence grants.

Any breach of this Licence by a Subsidiary shall entitle Arm to terminate this Licence as if you were the party in breach. Any termination of this Licence shall be effective in respect of all Subsidiaries. Any rights granted to any Subsidiary hereunder shall automatically terminate upon such Subsidiary ceasing to be a Subsidiary.

The Document consists solely of commercial items. Licensee shall be responsible for ensuring that any use, duplication or disclosure of the Document complies fully with any relevant export laws and regulations to assure that the Document or any portion thereof is not exported, directly or indirectly, in violation of such export laws.

This Licence may be translated into other languages for convenience, and Licensee agrees that if there is any conflict between the English version of this Licence and any translation, the terms of the English version of this Licence shall prevail.

The validity, construction and performance of this Licence shall be governed by English Law.

Arm Limited. Company 02557590 registered in England.

110 Fulbourn Road, Cambridge, England CB1 9NJ.

LES-PRE-21585 version 4.0

About this document

Terms and abbreviations

Term	Meaning
ACPI	The Advanced Configuration and Power Interface specification. This defines a standard for device configuration and power management by an OS
CoreSight	The CoreSight architecture provides a system-wide solution for real-time debug and collecting trace information

References

This section lists publications by Arm and by third parties.

See Arm Developer (http://developer.arm.com) for access to Arm documentation.

[1] *ARM CoreSight Architecture Specification v3.0*. See https://developer.arm.com/documentation/ihi0029/latest

[2] *Device Graphs. Using _DSD to represent arbitrary graphs DSD based graphs. UEFI Forum*. See https://github.com/UEFI/DSD-Guide

[3] *Arm® Architecture Reference Manual for A-profile architecture*. See https://developer.arm.com/documentation/ddi0487/latest

[4] *ARM CoreSight Trace Memory Controller Revision r0p1 Technical Reference Manual*. See https://developer.arm.com/documentation/ddi0461/b/

[5] *Arm CoreSight System-on-Chip SoC-600 Technical Reference Manual*. See https://developer.arm.com/documentation/100806/0200

[6] *Advanced Configuration and Power Interface Specification. UEFI Forum*. See http://uefi.org/specifications

Feedback

Arm welcomes feedback on its documentation.

If you have comments or suggestions for additions and improvements, please create a ticket at https://support.developer.arm.com. As part of the ticket please include:

- The title ACPI for CoreSight™.
- The number DEN0067 1.2.
- The section name to which your comments refer.
- The page number(s) that your comments apply to.
- A concise explanation of your comments.

Arm also welcomes general suggestions for additions and improvements.

Note

Arm tests PDFs only in Adobe Acrobat and Acrobat Reader, and cannot guarantee the appearance or behavior of any document when viewed with any other PDF reader.

Inclusive terminology commitment

Arm values inclusive communities. Arm recognizes that we and our industry have used terms that can be offensive. Arm strives to lead the industry and create change.

Previous issues of this document included terms that can be offensive. We have replaced those terms. If you find offensive terms in this document, please contact terms@arm.com.

1 ACPI description for CoreSight trace components

This specification describes how to support CoreSight [1] trace components with the Advanced Configuaration and Power Interface (ACPI). This specification is based on the ACPI _DSD graph specification [2], which provides support for representing system components that are arranged as a set of connected devices. This is the case with CoreSight, where components might be:

- trace sources, such as a CPU ETM trace unit or an STM
- trace sinks, such as an ETB
- both, such as funnels or replicators

The following sections describe:

- The CoreSight graph structure
- Device identifiers, _CID and _HID, for CoreSight components
- How to represent resources for CoreSight components
- Power management and CoreSight components
- Reference example

1.1 CoreSight graph structure

Each CoreSight component is described in the namespace using a device. The component type is described by a _CID, and individual implementations must use a _HID assigned by the vendor. The ID allocation for Arm IPs is described in Section 1.2.

A CoreSight device node must be declared as a child of the device that owns it. For CPUs, the CoreSight device nodes would typically be for ETM trace elements for that CPU. For devices other than CPUs, the device that is producing the trace data must also be declared in the namespace DSDT. CoreSight devices that are system-level, such as funnels or replicators, must be declared under the scope of the device that describes the system.

The ACPI _DSD device graph format [2] must be used to describe the graph topology of the CoreSight trace system.

ACPI _DSD graphs use a UUID to indicate the specification that governs the behavior of the graph. The specification UUID for CoreSight graphs is:

- *3ecbc8b6-1d0e-4fb3-8107-e627f805c6cd*

In addition to the rules for graphs that are imposed in [2], the following rules must be observed:

1. Link descriptors must include an additional field, called Direction, to indicate whether the originator of the link is a producer or a consumer. The format of this data is an integer. A value of 1 indicates producer, and a value of 0 indicates consumer. This property reflects the direction of data flow, and applies to the source port of a link. The source port is either an output port, in the producer case, or an input port, in the consumer case:

```
Package() { Source Port, Destination Port, Destination Device, Direction }
// Direction indicates whether source port is an output port on producer
// or input port on consumer
```

2. All output port numbers for a particular component must be unique.
3. All input port numbers for a given component must be unique.
4. A given source port cannot be used by more than one link. The port is identified by its number and by whether it is an output or input port. Thus, output port 0 and input port 0 on the same component are different ports.

5. Links must be declared in both components that are connected by that link. In the component that produces the data, the link declaration must have the direction field set to 1. This link is called the forward link. In the component that sinks the data, the link declaration must have the direction field set to 0. This link is called the backward link.

   For example:

   In the producer component, a forward link that targets the consumer component is described:

```
Device (PROD) { // Producer device
...
  Package() { 0, 1, CSMR, 1 } // Forward link declaration
  // Output port 0 is connected to
  // Input port 1 of the consumer device
  // Direction = 1 indicates that this link originates in the producer
}
```

   In the consumer component, a backward link that terminates in the consumer is described:

```
Device (CSMR) { // Consumer device
...
  Package() { 1, 0, PROD, 0 } // Backward link declaration
  // Input port 1 is connected to
  // Output port 0 of the producer device
  // Direction = 0 indicates that this link terminates in the consumer
}
```

1.2 Device identifier

Table 3 shows the compatible IDs that are associated with architected CoreSight components.

Table 3: Compatible IDs for architected CoreSight components

Component	Identifier
Coresight ETE [3] and CoreSight-ETMv4.x [4]	ARMH C500
CoreSight-ETR [4]	ARMH C501
CoreSight-STM [4]	ARMH C502
CoreSight-Debug	ARMH C503
CoreSight-Replicator-Static(*) [4]	ARMH C985
CoreSight-Funnel-Static(*) [4]	ARMH C9FE

(*) The term static denotes lack of an MMIO interface.

Table 4 shows the compatible IDs that are associated with Arm's CoreSight IP implementations.

Table 4: Hardware IDs for Arm's CoreSight IP implementations

Component	Identifier	Description
CoreSight-TMC [4]	ARMH C97C	This ID describes: • SoC 400 ETB • Coresight TMC configured as ETF, ETR or ETB when integrated with Coresight SoC 400 products • CoreSight SoC 600 TMC configured as ETF, ETB or ETS **Note**: CoreSight SoC 600 TMC ETR is covered by the ETR compatible ID described in Table 3.
CoreSight-Funnel [4]	ARMH C9FF	This ID covers all CoreSight funnels except for the static funnels that are described in Table 3.
CoreSight-TPIU [4]	ARMH C979	This ID covers CoreSight TPIU products.
CoreSight-Replicator [4]	ARMH C98D	This ID covers all CoreSight replicators except for the static replicators that are described in Table 3.
CoreSight-CATU [5]	ARMH C9CA	

1.3 Resources

Each CoreSight component must declare the resources that it owns using the _CRS method. This method must include the base address and span of the MMIO interface of the device, if the device cannot be accessed using system instructions. Otherwise, providing the base address and span is optional. Components which can raise interrupts must describe the interrupts they consume.

For STM, two base addresses must be presented, which must be provided in order. The first is the configuration base address, and the second is the the base address the external stimuli memory region.

1.4 Power

Where necessary, devices declared in the namespace to describe CoreSight components can use standard power methods (_PSx, _PRx). If _PR0 is implemented for a given device, the OSPM must ensure that the power resources it lists are in the ON state before the associated CoreSight component is used. Presenting a _PR0 also allows an OSPM to prevent entry into Lower Power Idle states that might turn off the resources associated with the CoreSight component, if the DSDT supplies _LPI and _RDI methods for those resources. Equivalently, if _PS0 is implemented, the OSPM must invoke the method before the associated CoreSight component is used.

1.5 Example

Consider the example system shown in the following figure:

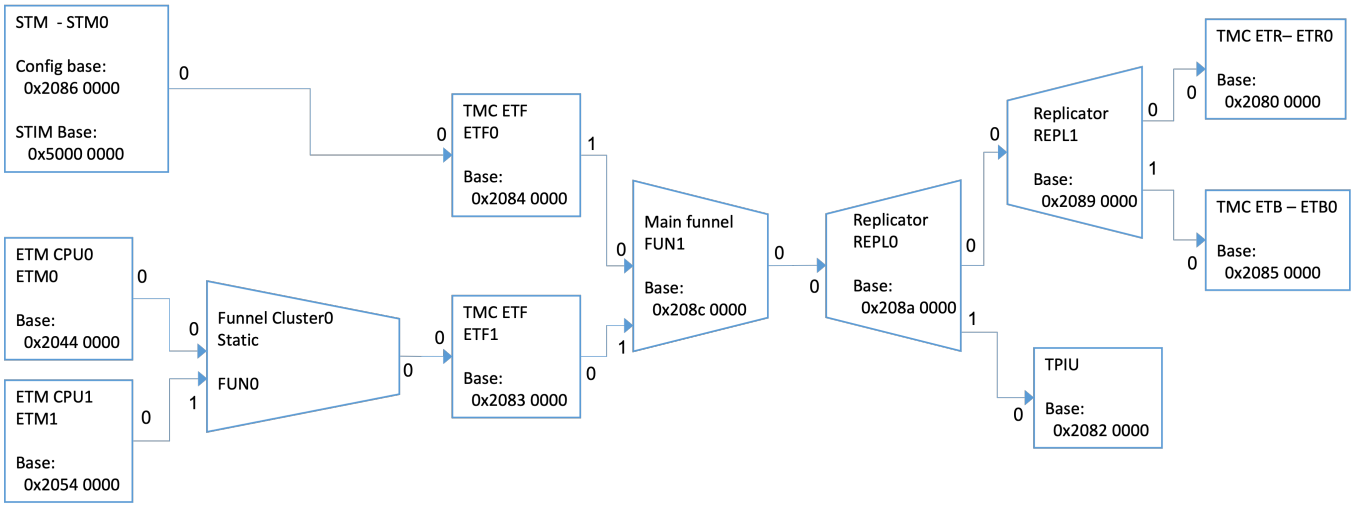

Figure 1: Example system

This example system can be described by the following ASL code:

```
Scope(\_SB) {

  Device (CLU0) { // Cluster0 state
    Name(_HID, "ACPI0010")
    ...
    Device (CPU0) { // CPU0
      Name(_HID, "ACPI0007")
      ...
      Device (ETM0) { // ETM on CPU0
        Name (_HID, "ARMHC500")
        Name(_CRS, ResourceTemplate () {
          Memory32Fixed(ReadWrite, 0x20440000, 0x1000)
        })

        Name (_DSD, Package () {
          ToUUID("ab02a46b-74c7-45a2-bd68-f7d344ef2153"),
          Package () {
            0,    // Revision
            1,    // Number of graphs
            Package () {
              1, // GraphID // CoreSight graph UUID
              ToUUID("3ecbc8b6-1d0e-4fb3-8107-e627f805c6cd"),
              1, // Number of links

              // Forward link between ETM0 and FUN0
              Package() {0, 0,       // output port 0 to connected
                  \_SB.CLU0.FUN0, 1}  // to input port 0 on FUN0

          }
        })
    ...
    }

    Device (CPU1) { // CPU1
      Name(_HID, "ACPI0007")
      ...
```

```
    Device (ETM1) { // ETM on CPU0
      Name (_HID, "ARMHC500")
      Name(_CRS, ResourceTemplate () {
        Memory32Fixed(ReadWrite, 0x20540000, 0x1000)
      })

      Name (_DSD, Package () {
        ToUUID("ab02a46b-74c7-45a2-bd68-f7d344ef2153"),
        Package () {
          0,    // Revision
          1,    // Number of graphs
          Package () {
            1, // GraphID // CoreSight graph UUID
            ToUUID("3ecbc8b6-1d0e-4fb3-8107-e627f805c6cd"),
            1, // Number of links

            // Forward link between ETM1 and FUN0
            Package() {0, 1,        // output port 0 to connected
                \_SB.CLU0.FUN0, 1}   // to input port 1 on FUN0
        }
      })
    }
  ...
  } // End of CPU1

  Device (FUN0) { // Funnel 0 described in cluster 0 scope
    Name (_HID, "ARMHC9FF")
    Name (_CID, "ARMHC500")

    Name (_DSD, Package () {
      ToUUID("ab02a46b-74c7-45a2-bd68-f7d344ef2153"),
      Package () {
        0,    // Revision
        1,    // Number of graphs
        Package () {
           1, // GraphID // CoreSight graphs UUID
           ToUUID("3ecbc8b6-1d0e-4fb3-8107-e627f805c6cd"),
           3, // Number of links

           // Backward link between FUN0 and ETM0
           Package() {0, 0,              // input port 0 connected
                \_SB.CLU0.CPU0.ETM0, 0}, // to output port 0 on ETM0

           // Backward link between FUN0 and ETM1
           Package() {1, 0,              // input port 1 to connected
                \_SB.CLU0.CPU1.ETM1, 0}, // to output  port 0 on ETM1

          // Forward link between FUN0 and ETF1
           Package() {0, 0,             // output port 0 connected
                \_SB.ETF1, 1}           // to input port 0 on ETF1
        }
      })
    }

...
}  // end of cluster0


Device (ETF1) { // ETF at 0x20830000 described \SB scope
  Name (_HID, "ARMHC97C")
```

```
  Name (_CID, "ARMHC500")
  Name(_CRS, ResourceTemplate () {
    Memory32Fixed(ReadWrite, 0x20830000, 0x1000)
  })

  Name (_DSD, Package () {
    ToUUID("ab02a46b-74c7-45a2-bd68-f7d344ef2153"),
    Package () {
      0,    // Revision
      1,    // Number of graphs
      Package () {
         1, // GraphID // CoreSight graphs UUID
         ToUUID("3ecbc8b6-1d0e-4fb3-8107-e627f805c6cd"),
         2, // Number of links

         // Forward link between ETF1 and FUN1
         Package() {0, 1,        // output port 0 connected
              \_SB.FUN1, 1},     // to input port 1 on FUN1.

        // Backward link between ETF1 and FUN0
         Package() {0, 0,         // input port 0 connected
              \_SB.CLU0.FUN0, 0}  // to output port 0 on FUN0.
      }
    }
  })
}



Device (STM0) { // STM0
  Name (_CID, "ARMHC502")  // STM
  Name(_CRS, ResourceTemplate () {
    Memory32Fixed(ReadWrite, 0x20860000, 0x1000)
    Memory32Fixed(ReadWrite, 0x50000000, 0x1800000) // stimulus
  })

  Name (_DSD, Package () {
    ToUUID("ab02a46b-74c7-45a2-bd68-f7d344ef2153"),
    Package () {
      0,    // Revision
      1,    // Number of graphs
      Package () {
         1, // GraphID // CoreSight graphs UUID
         ToUUID("3ecbc8b6-1d0e-4fb3-8107-e627f805c6cd"),
         1, // Number of links

         // Forward link between STM0 and ETF0
         Package() {0, 0,    // output port 0 connected
              \_SB.ETF0, 1}  // to output port 0 on ETF0.
      }
    }
  })
}


Device (ETF0) { // ETF at 0x20840000 described \SB scope
  Name (_HID, "ARMHC97C")
  Name (_CID, "ARMHC500")
  Name(_CRS, ResourceTemplate () {
    Memory32Fixed(ReadWrite, 0x20840000, 0x1000)
```

```
  })

  Name (_DSD, Package () {
    ToUUID("ab02a46b-74c7-45a2-bd68-f7d344ef2153"),
    Package () {
      0,    // Revision
      1,    // Number of graphs
      Package () {
         1, // GraphID // CoreSight graphs UUID
         ToUUID("3ecbc8b6-1d0e-4fb3-8107-e627f805c6cd"),
         2, // Number of links

         // Forward link between ETF0 and FUN1
         Package() {0, 0,    // output port 0 connected
              \_SB.FUN1, 1}, // to input port 0 on FUN1.

         // Backward link between ETF0 and STM0
         Package() {0, 0,    // input port 0 connected
              \_SB.STM0, 0}  // to output port 0 on STM0.
       }
    }
  })
}


Device (FUN1) { // Funnel 1 described in \SB scope
  Name (_HID, "ARMHC9FF")
  Name (_CID, "ARMHC500")
  Name(_CRS, ResourceTemplate () {
    Memory32Fixed(ReadWrite, 0x208c0000, 0x1000)
  })

  Name (_DSD, Package () {
    ToUUID("ab02a46b-74c7-45a2-bd68-f7d344ef2153"),
    Package () {
      0,    // Revision
      1,    // Number of graphs
      Package () {
         1, // GraphID // CoreSight graphs UUID
         ToUUID("3ecbc8b6-1d0e-4fb3-8107-e627f805c6cd"),
         3, // Number of links

         // Forward link between FUN1 and RPL0
         Package() {0, 0,    // output port 0 connected
              \_SB.RPL0, 1}, // to input port 0 on RPL0.

         // Forward link between FUN1 and ETF0
         Package() {0, 0,    // input port 0 connected
              \_SB.ETF0, 0}, // to output port 0 on ETF0.

         // Backward link between FUN1 and ETF1
         Package() {1, 0,    // input port 1 connected
              \_SB.ETF1, 0}  // to output port 0 on ETF1.
       }
    }
  })
}

Device (RPL0) { // Replicator 0 described in \SB scope
  Name (_HID, "ARMHC98D")
```

```
  Name (_CID, "ARMHC502")
  Name(_CRS, ResourceTemplate () {
    Memory32Fixed(ReadWrite, 0x208a0000, 0x1000)
  })

  Name (_DSD, Package () {
    ToUUID("ab02a46b-74c7-45a2-bd68-f7d344ef2153"),
    Package () {
      0,    // Revision
      1,    // Number of graphs
      Package () {
         1, // GraphID // CoreSight graphs UUID
         ToUUID("3ecbc8b6-1d0e-4fb3-8107-e627f805c6cd"),
         3, // Number of links

         // Forward link between RPL0 and RPL1
         Package() {0, 0,    // output port 0 connected
              \_SB.RPL1, 1}, // to input to port 0 on RPL1.

         // Forward link between RPL0 and TPIU
         Package() {1, 0,    // output port 1 connected
              \_SB.TPIU, 1}, // to input port 0 on TPIU.

         // Backward link between RPL0 and FUN1
         Package() {0, 0,    // input port 0 connected to
              \_SB.FUN1, 0}  // to output port 0 on FUN1.
       }
    }
  })

}

Device (TPIU) { // TPIU described in \SB scope
  Name (_HID, "ARMHC979")
  Name (_CID, "ARMHC501")
  Name(_CRS, ResourceTemplate () {
    Memory32Fixed(ReadWrite, 0x20820000, 0x1000)
  })


  Name (_DSD, Package () {
    ToUUID("ab02a46b-74c7-45a2-bd68-f7d344ef2153"),
    Package () {
      0,    // Revision
      1,    // Number of graphs
      Package () {
         1, // GraphID // CoreSight graphs UUID
         ToUUID("3ecbc8b6-1d0e-4fb3-8107-e627f805c6cd"),
         1, // Number of links

         // Forward link between TPIU and RPL0
         Package() {0, 1,    // input port 0 connected
              \_SB.RPL0, 0}  // to output  port 1 on RPL0.
       }
    }
  })
}

Device (RPL1) { // Replicator 1 described in \SB scope
  Name (_HID, "ARMHC98D")
```

```
  Name (_CID, "ARMHC502")
  Name(_CRS, ResourceTemplate () {
    Memory32Fixed(ReadWrite, 0x20890000, 0x1000)
  })

  Name (_DSD, Package () {
    ToUUID("ab02a46b-74c7-45a2-bd68-f7d344ef2153"),
    Package () {
      0,    // Revision
      1,    // Number of graphs
      Package () {
         1, // GraphID // CoreSight graphs UUID
         ToUUID("3ecbc8b6-1d0e-4fb3-8107-e627f805c6cd"),
         3, // Number of links

         // Forward link between RPL1 and ETR0
         Package() {0, 0,    // output port 0 connected
              \_SB.ETR0, 1}, // to input port 0 on ETR0.

         // Forward link between RPL1 and ETB0
         Package() {1, 0,    // output port 1 connected
              \_SB.ETB0, 1}, // to input port 0 on ETB0.

         // Backward link between RPL1 and RPL0
         Package() {0, 0,    // input port 0 connected
              \_SB.RPL0, 0}  // to output port 0 on RPL0.
       }
    }
  })
}

Device (ETR0) { // ETR0 described in \SB scope
  Name (_CID, "ARMHC501")
  Name(_CRS, ResourceTemplate () {
    Memory32Fixed(ReadWrite, 0x208000000, 0x1000)
  })

  Name (_DSD, Package () {
    ToUUID("ab02a46b-74c7-45a2-bd68-f7d344ef2153"),
    Package () {
      0,    // Revision
      1,    // Number of graphs
      Package () {
         1, // GraphID // CoreSight graphs UUID
         ToUUID("3ecbc8b6-1d0e-4fb3-8107-e627f805c6cd"),
         1, // Number of links

         // Backward link between ETR0 and RPL1
         Package() {0, 0,    // input port 0 connected
              \_SB.RPL1, 0}  // to output port 0 on RPL1.
       }
    }
  })
}

Device (ETB0) { // ETB0 described in \SB scope
  Name (_HID, "ARMHC97C")
  Name (_CID, "ARMHC500")
  Name(_CRS, ResourceTemplate () {
    Memory32Fixed(ReadWrite, 0x20850000, 0x1000)
```

```
    })

    Name (_DSD, Package () {
      ToUUID("ab02a46b-74c7-45a2-bd68-f7d344ef2153"),
      Package () {
        0,    // Revision
        1,    // Number of graphs
        Package () {
           1, // GraphID // CoreSight graphs UUID
           ToUUID("3ecbc8b6-1d0e-4fb3-8107-e627f805c6cd"),
           1, // Number of links

           // Backward link between ETB0 and RPL1
           Package() {0, 1,    // input port 0 connected
                \_SB.RPL1, 0}  // to output port 1 on RPL1.
         }
      }
  }
...
}
```