AN12239

How to Enable HyperRAM with i.MX RT

Rev. 2 — 7 May, 2021 Application Note

Contents

1 Introduction

The i.MX RT series MCU is a crossover product from NXP. It includes a Flexible Serial Peripheral Interface (FlexSPI) controller which supports HyperBus devices (HyperFlash/HyperRAM). This application note describes how to use the HyperRAM with the i.MX RT MCU, including hardware connections, HyperRAM protocol, source code, and performance.

The SDK used for the example in this application note is SDK_2.3.1_EVKB-IMXRT1050. The development environment is IAR Embedded Workbench® 8.22.1 IDE. The hardware environment is the MIMXRT1050-EVKB board. The HyperRAM chip is S27KS0641 from Cypress®.

2 MIMXRT1050 EVK board setting

By default, the HyperFlash chip (Cypress S26KS512SDPBHI02) is connected to the FlexSPI interface on the MIMXRT1050-EVKB board. The HyperFlash chip (as shown in Figure 1) is replaced with the HyperRAM.

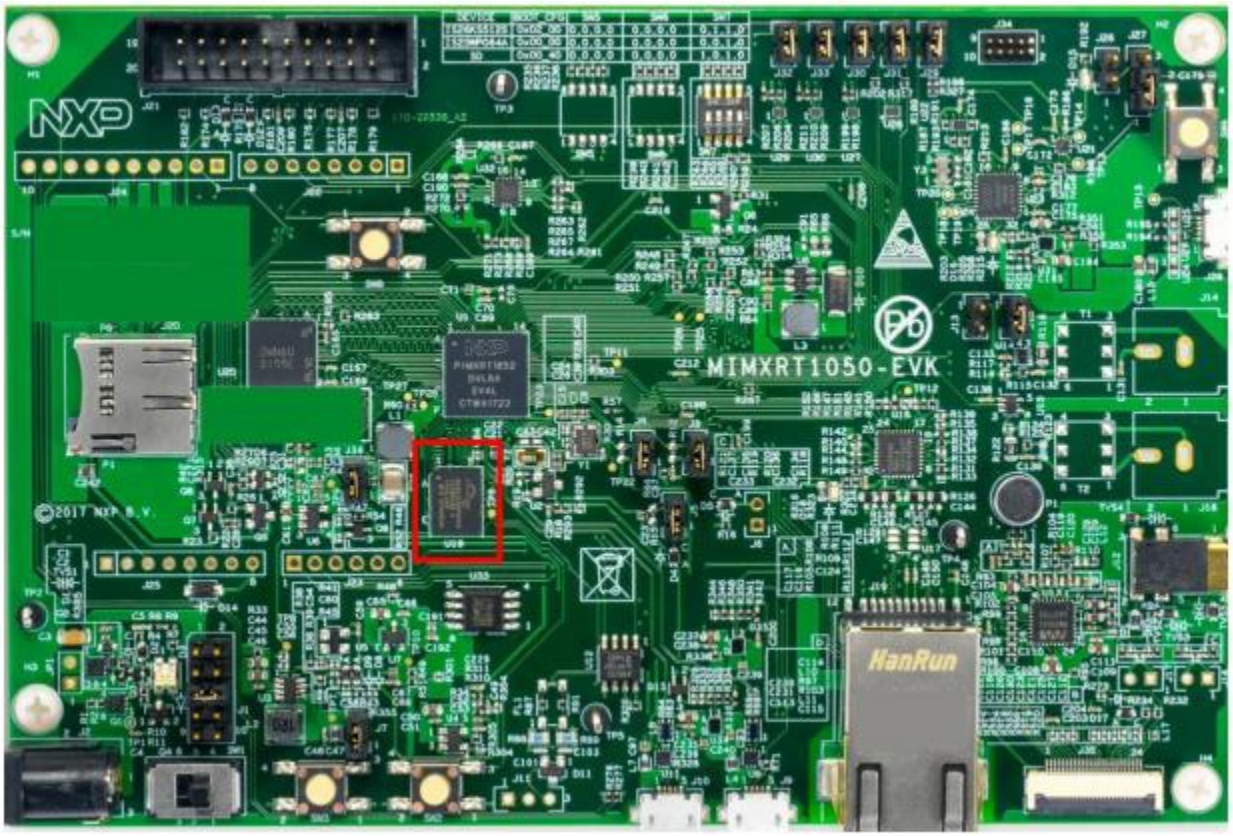

Figure 1. Replacing HyperFlash with HyperRAM

2.1 Board re-work for HyperRAM device

The Cypress S27KS0641 HyperRAM has the same package as the default on-board Cypress S26KS512SDPBHI02 HyperFlash. The devices can be simply swapped without any other hardware changes required. Figure 2 shows the detailed PIN information for the replacement.

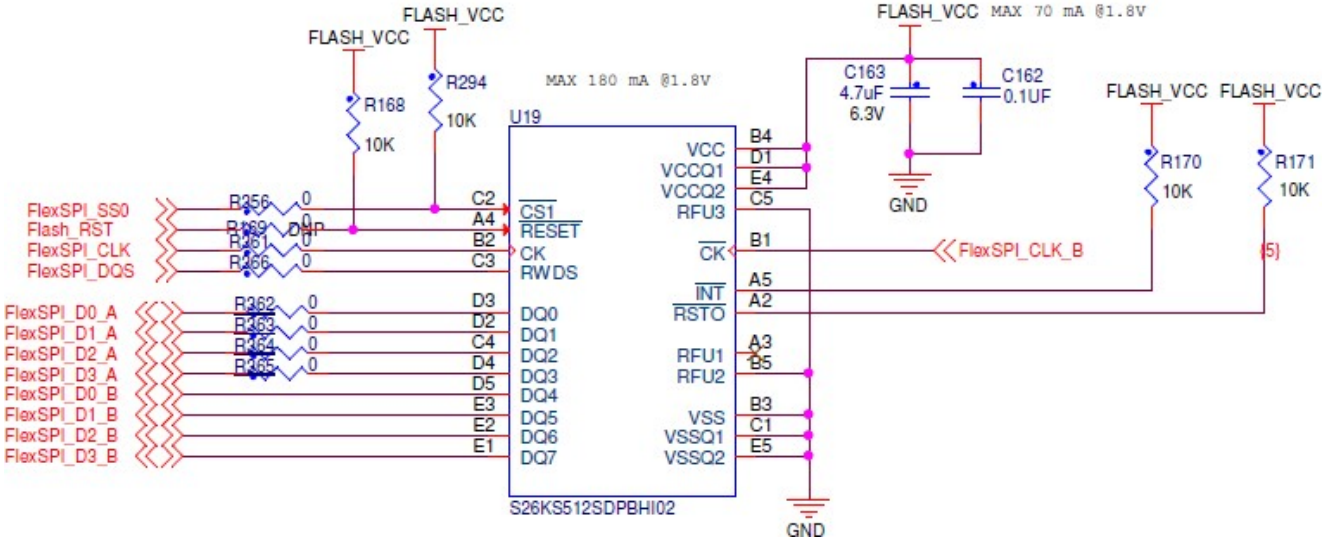

Figure 2. HyperRAM device rework

2.2 HyperRAM device

The Cypress S27KS0641 HyperRAM device features:

- 64-Mb (8-MB) self-refresh DRAM.
- 3.0-V I/O, 11 bus signals (CK); 1.8-V I/O, 12 bus signals (differential clock (CK, CK#)).
- 166-MHz clock rate (333 MB/s) at 1.8-V VCC; 100-MHz clock rate (200 MB/s) at 3.0-V VCC.
- Double-Data Rate (DDR)—two data transfers per clock.
- 8-bit data bus (DQ[7:0]).
- Read-Write Data Strobe (RWDS).
- Sequential burst transactions.
- Configurable burst characteristics.
- Low-power modes.
- 24-ball FBGA package.

Table 1 shows the Cypress S27KS0641 HyperRAM signal descriptions.

Table 1. S27KS0641 HyperRAM signal descriptions

Symbol	Type	Description
CS#	Master Output Slave Input	**Chip Select** Bus transactions are initiated with a HIGH to LOW transition. Bus transactions are terminated with a LOW to HIGH transition. The master device has a seperate CS# for each slave.

Table continues on the next page...

Table 1. S27KS0641 HyperRAM signal descriptions (continued)

Symbol	Type	Description
CK, CK#	Master Output Slave Input	**Differential Clock** Command, address, and data information is output with respect to the crossing of the CK and CK# signals. Differential clock is used on 1.8 V I/O devices. **Single Ended Clock** CK# is not used on 3.0 V devices. Only a single ended CK is used. The clock is not required to be free-running.
DQ[7:0]	Input/Output	**Data Input/Output** Command, Address, and Data information is transferred on these signals during Read and Write transaction.
RWDS	Input/Output	**Read Write Data Strobe** During the Command/Address portion of all bus transactions RWDS is a slave output and indicates whether addistional initial latency is required. Slave output during read data transfer, data is edge aligned with RWDS. Slave input during data transfer in write transactions to function as a data mask. (HIGH = additional latency, LOW = no additional latency)
RESET#	Master Output Slave Input Internal Pull-up	**Hardware RESET** When LOW, the slave device will self initialize and return to the Standby state. RWDS and DQ[7:0] are placed into the HI-Z state when RESET# is LOW. The slave RESET# input includes a weak pull-up. If RESET# is left unconnected, it will be pulled up to the HIGH state.
V_{cc}	Power Supply	**Power**
$V_{cc}Q$	Power Supply	**Input/Output Power**
V_{ss}	Power Supply	**Ground**
$V_{ss}Q$	Power Supply	**Input/Output Ground**
RFU	No Connect	**Reserved for Future Use** May or may not be connected internally. The signal/ball location should be left unconnected and unused by PCB routing channel for future compatibility. The signal/ball may be used by a signal in the future.

For more information about the Cypress S27KS0641 HyperRAM, see the Datasheet.

3 FlexSPI controller and HyperBus

3.1 FlexSPI host controller

FlexSPI is a flexible SPI host controller which supports two SPI channels and up to four external devices. Each channel supports the single/dual/quad modes of data transfer (1/2/4 bi-directional data lines). On the i.MX RT1050, the octal mode is supported by combining SIOA[3:0] and SIOB[3:0]. Figure 3 shows the block diagram of the FlexSPI host controller.

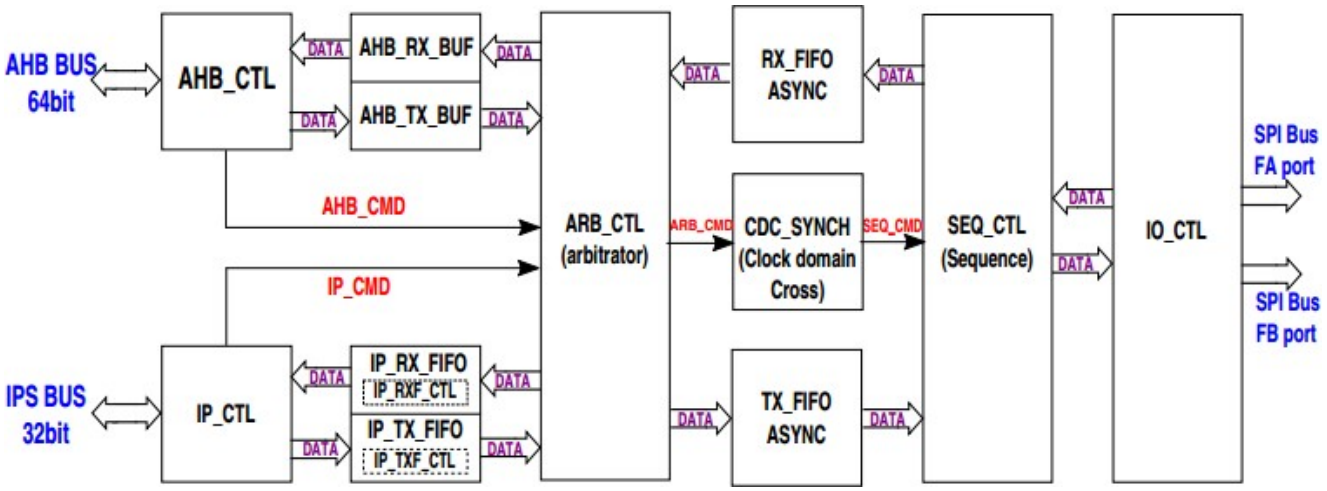

Figure 3. FlexSPI block

The FlexSPI host controller features:

- Flexible sequence engine (LUT table) to support various vendor devices.
- Flash access modes: single/dual/quad/octal, SDR/DDR, and individual/parallel.
- Read strobe clock sampling.
- Memory-mapped read/write access by the AHB bus:
  - The AHB RX buffer is implemented to reduce read latency. The total AHB RX buffer size is 128 × 64 bits.
  - The AHB TX buffer is implemented to buffer all write data from one AHB burst. The AHB TX buffer size is 8 × 64 bits.
- Software-triggered flash read/write access by the IP bus:
  - The IP RX FIFO is implemented to buffer all read data from the external device. Its size is 16 × 64 bits.
  - IP TX FIFO is implemented to buffer all write data to the external device. Its size is 16 × 64 bits.

3.2 HyperBus protocol

HyperBus has a low signal count and the Double Data Rate (DDR) interface, which achieves high read and write throughput while reducing the number of device I/O connections and signal routing congestion in a system.

The HyperBus interface features:

- 3.0-V I/O, 11 bus signals, single-ended clock (CK).
- 1.8-V I/O, 12 bus signals, differential clock (CK, CK#).
- Chip Select (CS#).
- 8-bit data bus (DQ[7:0]).
- Read-Write Data Strobe (RWDS).
- Double-Data Rate (DDR)—two data transfers per clock.
- Up to 200-MHz clock rate (400 MB/s) at 1.8-V/3.0-V VCC.
- Sequential burst transactions, configurable burst characteristics.

For the HyperBus protocol, the first three clock cycles transfer three words (48 bits in total) of the command/address (CA0, CA1, CA2) information to define the transaction characteristics. The command/address words are presented with the DDR timing, using the first six clock edges. The characteristics are defined by the command/address information in Table 2. Figure 4 shows the clock sequence of the command/address words.

Table 2. CA bit characteristics

CA Bit#	Bit Name	Bit function
47	R/W#	Identifies the transaction as Read or Write. R/W#=1 indicates a Read transaction. R/W#=0 indicates a Write transaction.
46	Address Space (AS)	Indicates whether the read or write transaction accesses the memory or register space. AS=0 indicates the memory space. AS=1 indicates the register space. The register space is used to access device ID and Configuration registers.
45	Burst Type	Indicates whether the burst will be linear or wrapped. Burst Type = 0 indicates wrapped burst. Burst Type = 1 indicates linear burst.
44-16	Row & Upper Column Address	Row & Upper Column component of the target address: System word address bits A31-A3. Any upper Row address bits not used by a particular device density should be set to 0 by the host controller master interface. The size of Rows and therefore the address bit boundary between Row and Column address is slave device dependent.
15-3	Reserved	Reserved for future column address expansion. Reserved bits are not cared in current HyperBus devices but should be set to 0 by the host controlller master interface for future compatibility.
2-0	Lower Column Address	Lower Column component of the target address: System word address bits A2-0 selecting the starting word within a half-page.

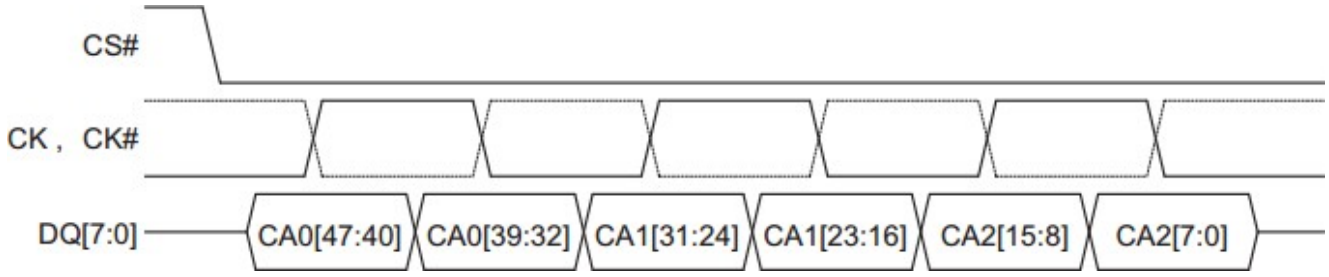

Figure 4. Command/address sequence

Figure 5 and Figure 6 show the HyperBus read/write clock sequence with the single initial latency count.

Figure 5 shows the read transactions with the single latency.

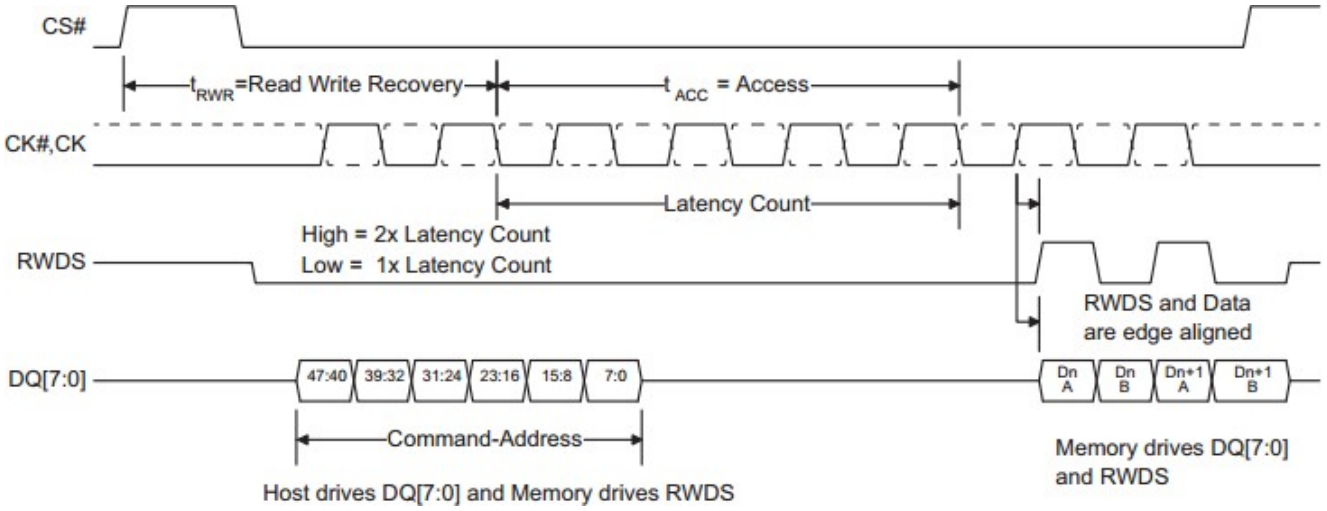

Figure 5. Read transaction, single initial latency count

Figure 6 shows the write transactions with the single latency.

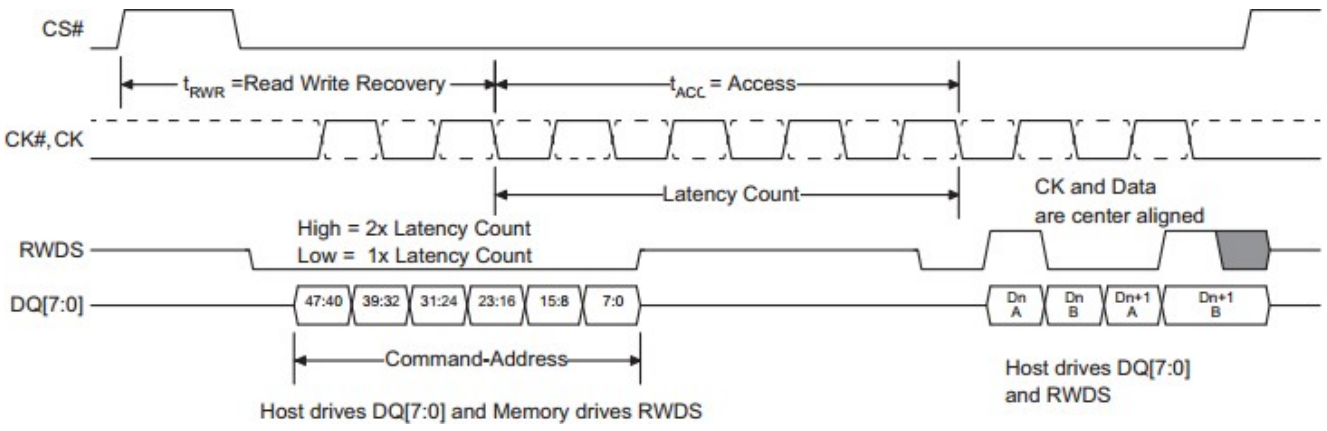

Figure 6. Write transaction, single initial latency count

For more information about the HyperBus protocol, see the Specifications.

4 Memory region and Look-Up-Table (LUT)

There are four key memory regions related to the FlexSPI controller and HyperRAM access in the i.MX RT1050 platform. The most important memory region is LUT.

4.1 FlexSPI register memory region

- Base address: `0x402A_8000h`
- Size: 16 KB

The FlexSPI controller register memory region includes all the configuration registers. The first step is to set the right FlexSPI controller mode, flash parameters, AHB/IP mode, LUT block, and so on, in the controller memory region.

4.2 AHB access memory region

- Base address: `0x6000_0000h`

- Size: 512 MB

The HyperBus device can be accessed by the AHB bus directly in the AHB address space of `0x60000000` - `0x80000000`. This address space is mapped in the serial flash/RAM memory in the FlexSPI. The AHB bus access to this address space triggers the flash/RAM access command sequence as needed.

For the AHB read access to the serial flash/RAM memory, the FlexSPI fetches the data from the flash/RAM to the AHB RX buffers and then returns the data to the AHB Bus. For the AHB write access to the serial flash/RAM memory, the FlexSPI buffers the write data from the AHB bus to the AHB TX buffers and then transmits it to the serial flash/RAM memory. There is no software configuration or polling needed for the AHB command except for the FlexSPI initialization.

The AHB bus access features:

- Cacheable and non-cacheable access for reading. When set to cacheable, the FlexSPI checks whether the reading address hit the AHB TX buffer first.
- Bufferable and non-bufferable access for writing.
- Prefetch enable/disable.
- Burst size: 8/16/32/64 bits.
- All burst types: SINGLE/INCR/WRAP4/INCR4/WRAP8/INCR8/WRAP16/INCR16.

4.3 IP command access memory region

- IP RX FIFO base address is:
  - `0x402A_8100h` – `0x402A_817Ch` (by IPS bus).
  - `0x7FC0_0000h` – `0x7FC0_007Ch` (by AHB bus).
- Size: 128 B

The FlexSPI puts the read data from the external device into the IP RX FIFO for the IP command. The data can be read out using either of the two above-mentioned memory spaces. `MCR0[ARDFEN]` defines the read memory space and method.

- IP TX FIFO base address is:
  - `0x402A_8180h` – `0x402A_81FCh` (by IPS bus).
  - `0x7F80_0000h` – `0x7F80_007Ch` (by AHB Bus).
- Size: 128 B

The write data should be put into the IP TX FIFO and then transmitted to the external device by the IP command. The data can be written into either of the two above-mentioned memory spaces.

`MCR0[ATDFEN]` defines the write memory space and method. The IP command access consists of these key steps:

1. Fill the IP TX FIFO with the write data if it is a write command.
2. Set the flash/RAM access start address (IPCR0), read/program data size, sequence index in LUT, and sequence number (IPCR1).
3. Trigger the flash access command by writing **1** to the register bit `IPCMD[TRG]`.
4. Poll the IPCMDDONE register bit to wait for the IP command to finish in the FlexSPI interface.

4.4 LUT memory region

- Base address: `0x402A_8200h`
- Size: 256 B

The LUT is an internal memory region to store a number of pre-programmed sequences. Each sequence consists of up to eight instructions which are executed sequentially. When a flash/RAM access is triggered by an IP command or an AHB command, the FlexSPI controller fetches the defined sequence from the LUT memory region according to the index/number values in the

configuration register and executes it to generate a valid flash/RAM transaction on the SPI interface. shows the structure of the LUT and the sequences and instructions.

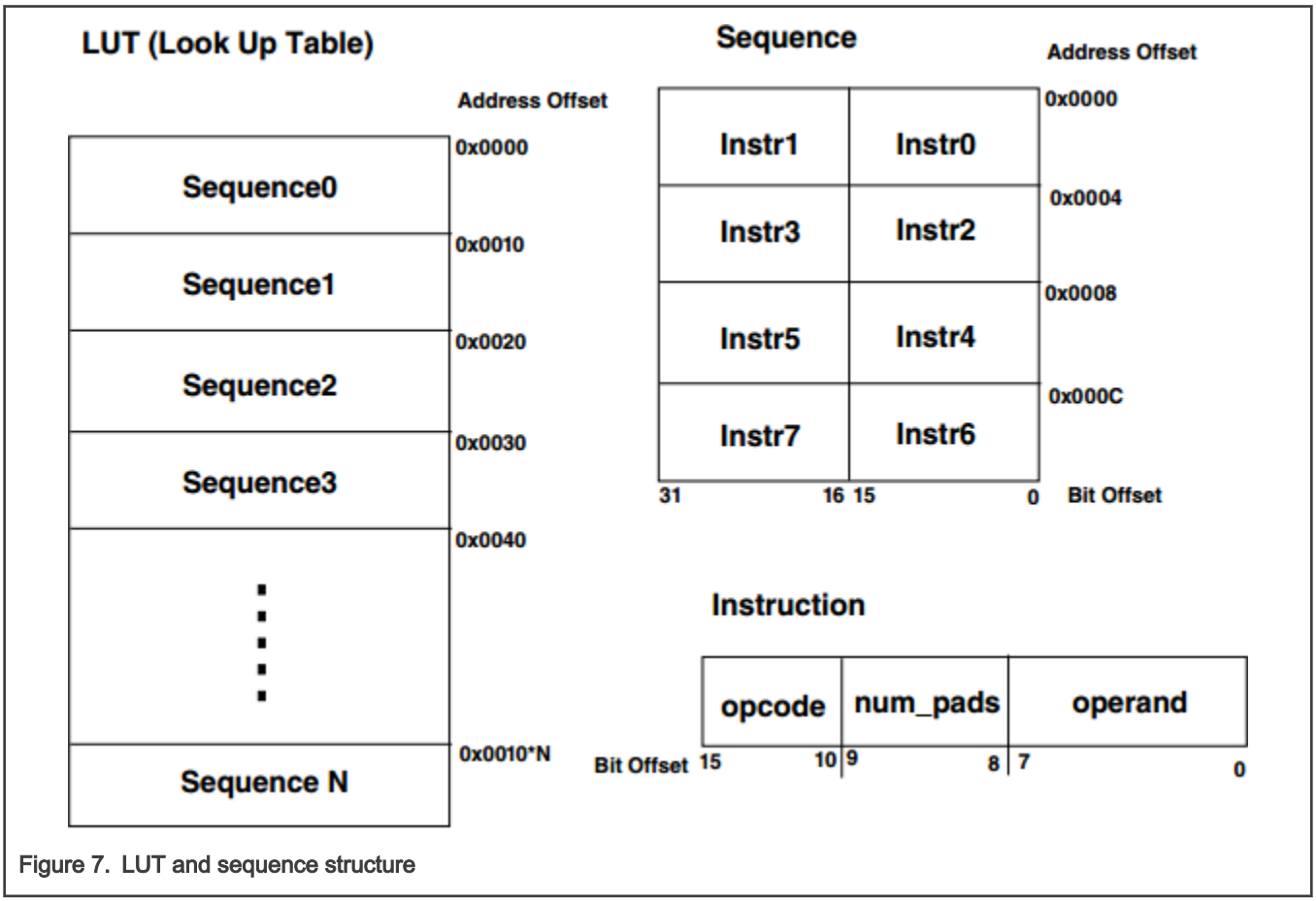

Figure 7. LUT and sequence structure

For detailed instruction information, see chapter 30.7.8 in *i.MX RT1050 Reference Manual* (document IMXRT1050RM).

5 Source code and performance

5.1 Running the HyperRAM example

The HyperRAM example source code is based on the i.MX RT1050 SDK V2.3.1. Download the code package *hyper_ram.zip* from NXP website.

1. Set up the hardware environment.

   Rework the MIMXRT1050 EVK board, as shown in MIMXRT1050 EVK board setting, to replace the Cypress S26KS512SDPBHI02 HyperFlash device with the Cypress S27KS0641 HyperRAM device. Then connect the OpenSDA/UART interface to the host PC and make sure that it powers on properly.

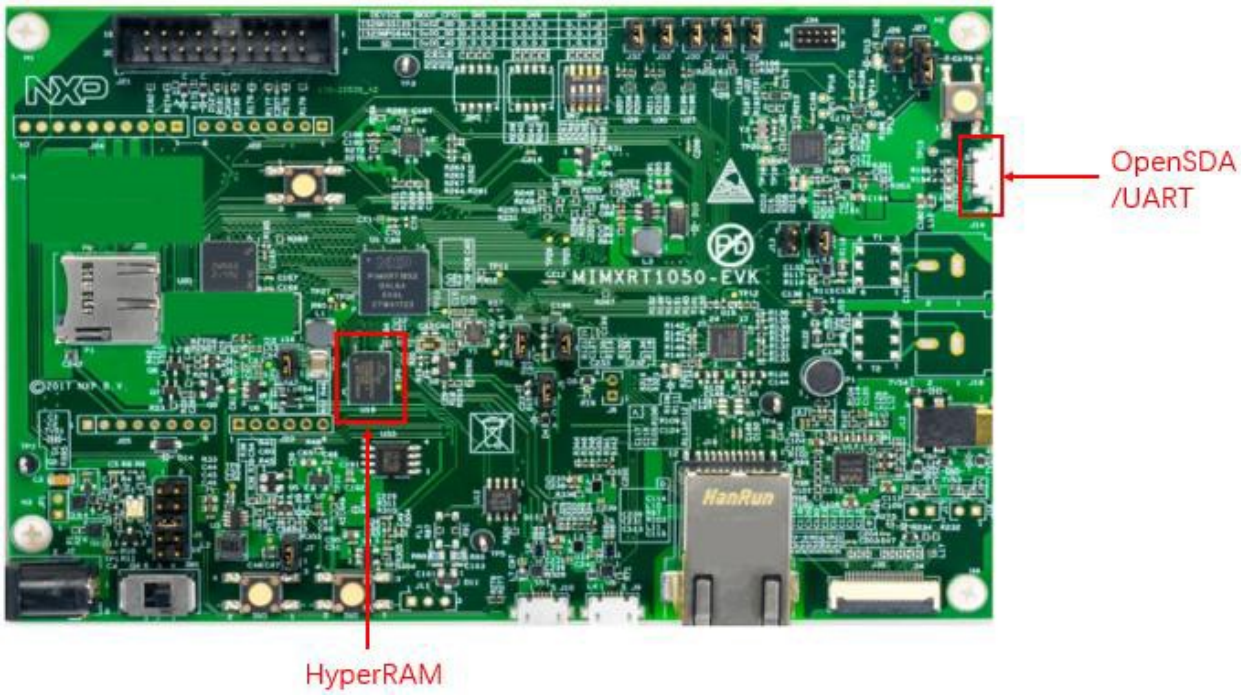

Figure 8. HyperRAM test board

2. Create the HyperRAM project.

   Unzip the *hyper_ram.zip* package and extract the source code of the HyperRAM example. Copy the *hyper_ram* folder into the *SDK_2.3.1_EVKB-IMXRT1050\boards\evkbimxrt1050\driver_examples\flexspi* folder of the i.MX RT1050 SDK V2.3.1.

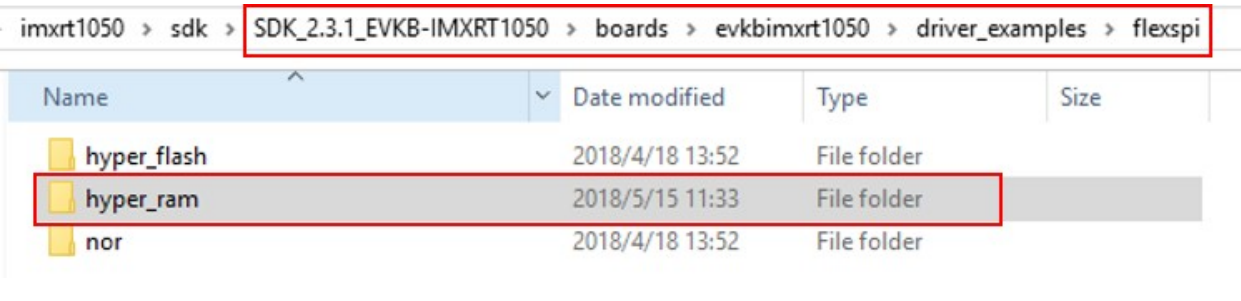

Figure 9. HyperRAM source code

3. Main blocks of the HyperRAM example code.

   In this example, the FlexSPI sends data and operates the external HyperRAM device connected to the FlexSPI interface.

   a. The example implements the necessary configurations of the i.MX RT1050 platform and configures the FlexSPI controller according to the HyperRAM device.

   b. The example implements the read/write operations from/to the HyperRAM device using the AHB and IP commands.

   c. A simple performance test is implemented and the results are displayed over the UART terminal connection. Double-click the *flexspi_hyper_ram_polling_transfer.eww* file to open the HyperRAM IAR project.

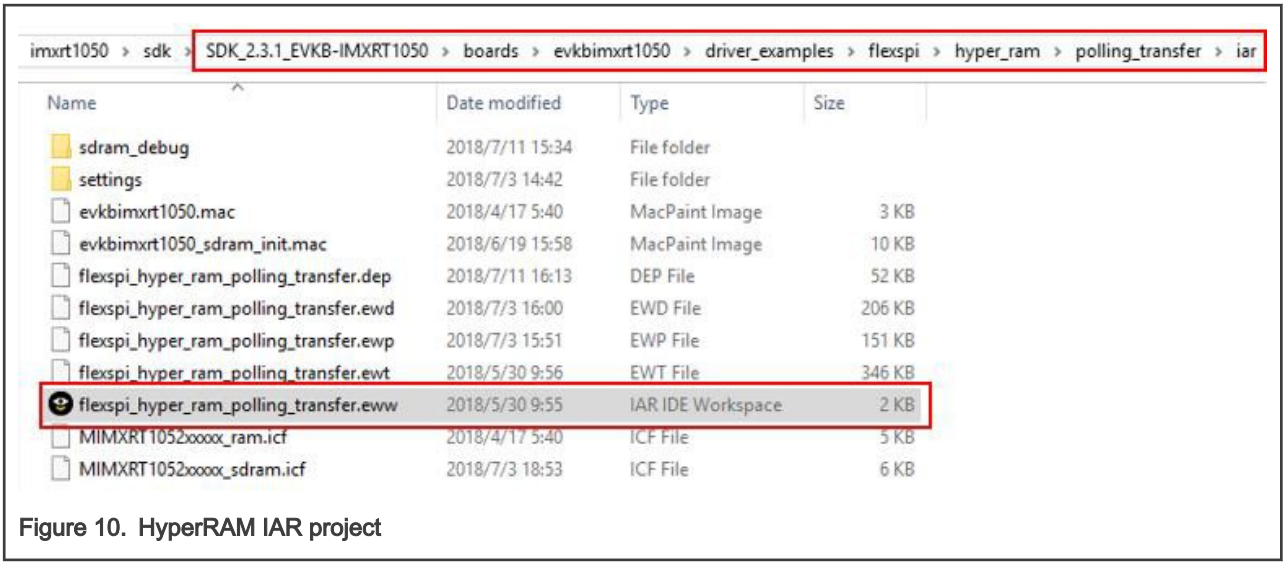

Figure 10. HyperRAM IAR project

4. Build and run the example.

   Find the MIMXRT1050 EVK OpenSDA-UART port on the host PC and open a serial terminal with these settings:

   - 115200 baud rate.
   - Eight data bits.
   - No parity.
   - One stop bit.
   - No flow control.

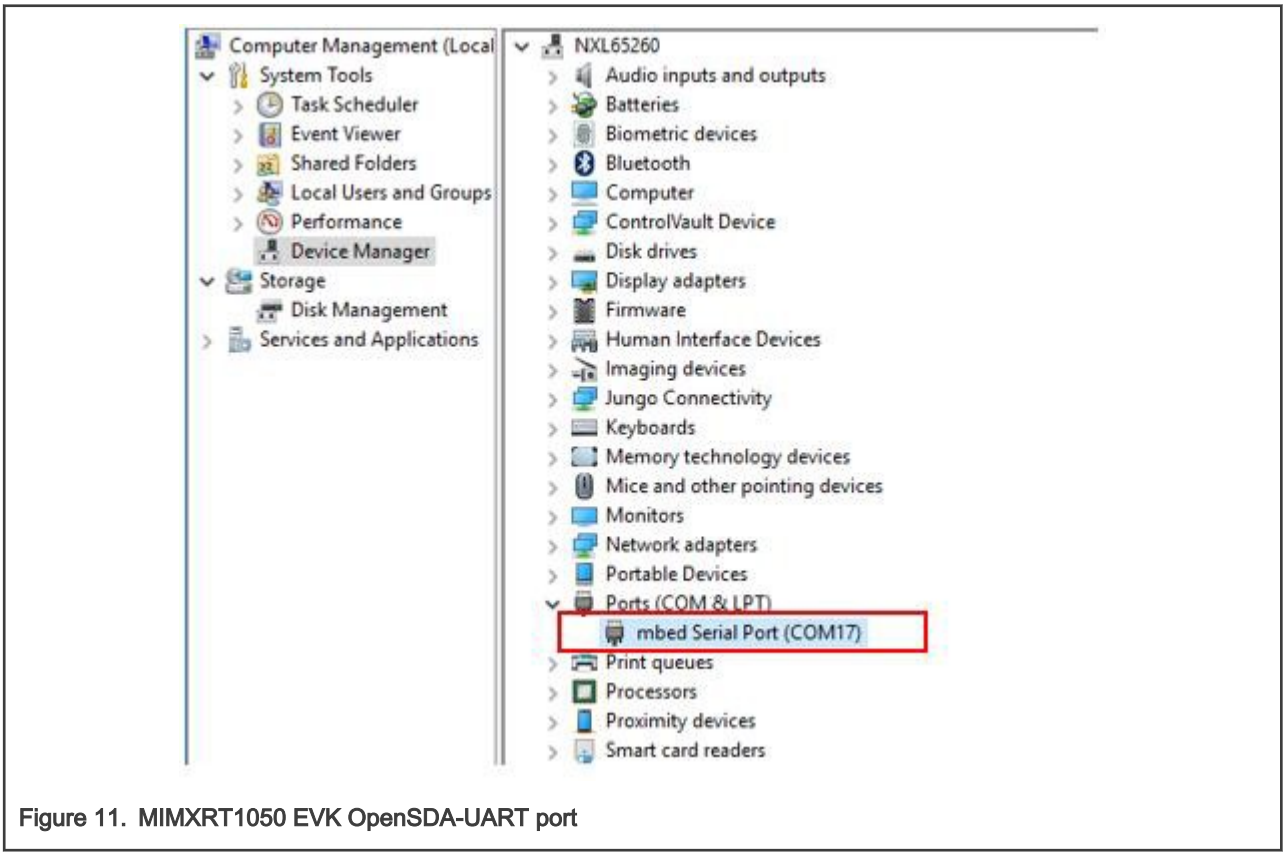

Figure 11. MIMXRT1050 EVK OpenSDA-UART port

5. Set the compiling optimizations level to **None**.

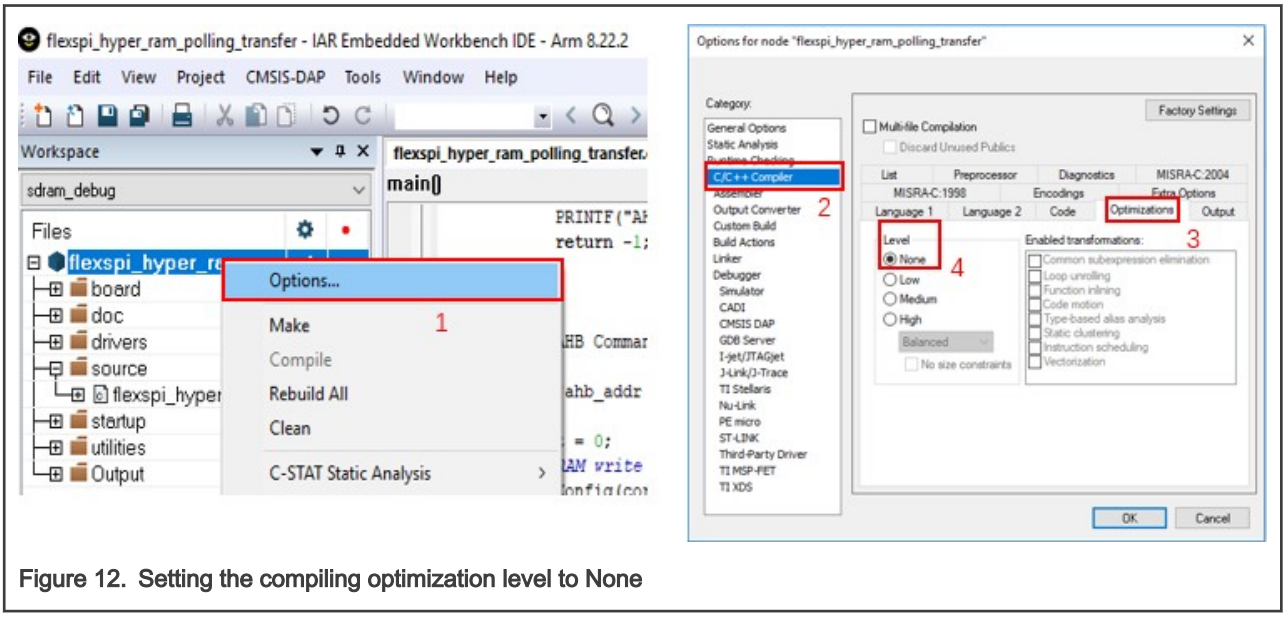

Figure 12. Setting the compiling optimization level to None

6. Make, download, and debug. The information shown in Figure 14 appears in the serial terminal.
   a. Select the **debug** project to make sure that the test data are put into the DTCM, and the code is put into the ITCM.
   b. Make the project.
   c. Download and debug.
   d. Click **Go** to run the project.

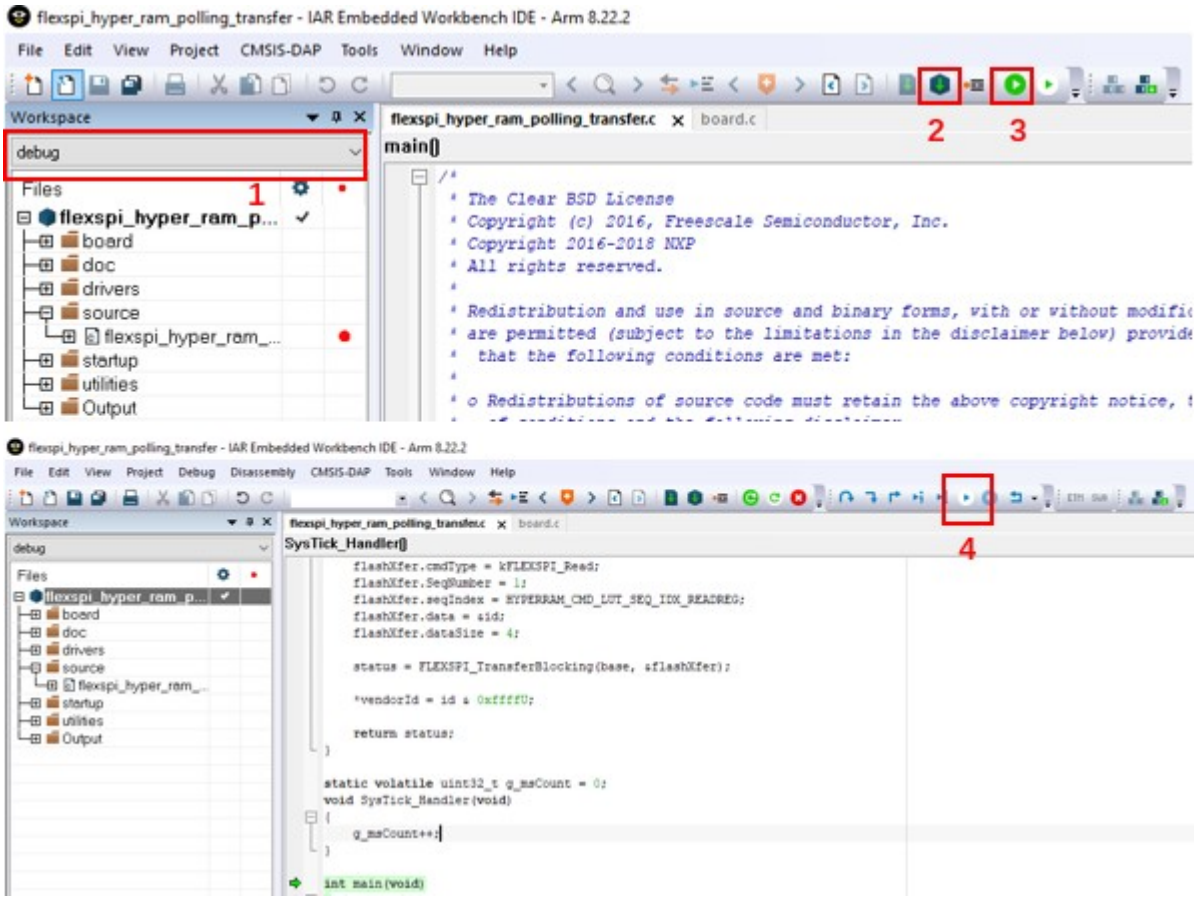

Figure 13. Building and running the HyperRAM project

```
coreclk: 600000000; ahbclk: 150000000; hyperbusclk: 332307684
FlexSPI HyperRAM example started!
Vendor ID: 0x810c
IP Command Read/Write data succeed at all address range !
AHB Command Read/Write data succeed at all address range !
##HyperRAM AHB write perf##t1: 599989; t2: 599920; diff: 69; ns: 115, datasize: 4 byte; perf: 34MB/s; g_ms: 0
##HyperRAM AHB read perf###t1: 599985; t2: 599932; diff: 53; ns: 88, datasize: 4 byte; perf: 45MB/s; g_ms: 0
0x60000000: 0x5a5a5a5a 0x 7060504 0x b0a0908 0x f0e0d0c
0x60000004: 0x 7060504 0x b0a0908 0x f0e0d0c 0x13121110
0x60000008: 0x b0a0908 0x f0e0d0c 0x13121110 0x17161514
0x6000000c: 0x f0e0d0c 0x13121110 0x17161514 0x1b1a1918
```

Figure 14. HyperRAM demo print information

5.2 Performance and analysis

The HyperRAM project described above includes the performance test cases. Table 3 shows the software configurations for the test.

Table 3. HyperRAM test environment

—	Module	Freq
Core	Arm® Cortex®-M7	600 MHz

Table continues on the next page...

Table 3. HyperRAM test environment (continued)

—	Module	Freq
AHB to FlexSPI	64-bit	150 MHz
IPS to FlexSPI	32-bit	150 MHz
HyperRAM chip	S27KS0641 @ 1.8 V	166 MHz (332 MB/s)
L1 Dcache	Total 32 KB/one-line 32 B	—
HyperRAM space setting	MPU: Normal memory type Non-shareable/cacheable/wb	—
FlexSPI controller setting	Read: Enable Prefetch Write: Enable Bufferable	—
Code	Text region in ITCM Data region in HyperRAM CStack region in DTCM Disable/enable Dcache Turn off compiling optimization	—

NOTE

The test cases in the HyperRAM project interact with each other. Deactivate the other accesses, except for the target case. For example, to test the HyperRAM read performance, deactivate the AHB access verification, IP access verification, and write performance cases.

There are two types of the performance test case: **Dcache disable** and **Dcache enable**.

- The **Dcache disable** test case shows the pure FlexSPI and HyperRAM performance. The analysis of the results expounds the mechanism to improve the performance and basic configurations.
- The **Dcache enable** test case shows how to promote the read performance via Dcache and why the performance can be promoted so significantly.

Following the above test cases, different configurations can be selected according to a specific use case. Table 4 and Figure 15 show the performance results with **Dcache disable**.

Table 4. HyperRAM core read/write performance with Dcache disabled

—	Byte	4	8	32	128	512	1024	16 K	32 K
Read	Time (ns)	278	338	696	2140	7896	15576	248876	497730
	Perf (MB/S)	14	23	45	59	64	65	65	65
Write	Time (ns)	115	121	188	1075	4648	9401	152155	304415
	Perf (MB/S)	34	66	170	119	110	108	107	107

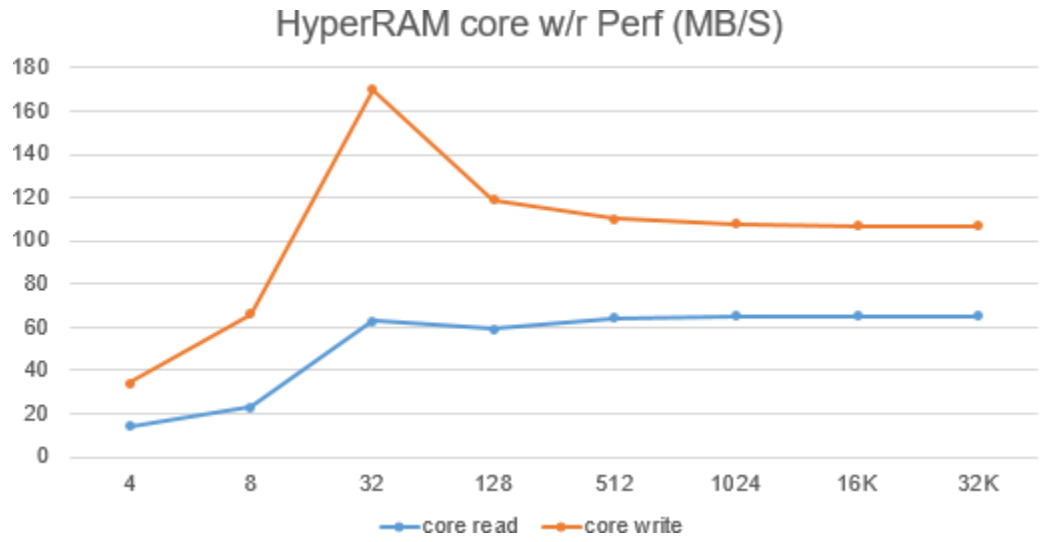

Figure 15. HyperRAM performance diagram with Dcache disabled

The analysis is based on the above-mentioned performance results.

- The HyperRAM memory space attributes defined in the MPU model:

  The original SDK code sets the HyperRAM memory space type to the **Device** mode. This limits the AHB burst write to the single mode and causes a very poor write performance.

  Changing the HyperRAM memory space type to the **Normal** mode in the MPU enables the AHB write bursts to the INCR, greatly improving the write performance.

  In the *board.c* file:

```
/* Setting Memory with Normal type, not shareable, outer/inner write back. */
MPU->RBAR = ARM_MPU_RBAR(2, 0x60000000U);
MPU->RASR = ARM_MPU_RASR(0, ARM_MPU_AP_FULL, 0, 0, 1, 1, 0, ARM_MPU_REGION_SIZE_512MB);
```

- Enable the FlexSPI write access bufferable feature:

  The FlexSPI optionally buffers the AHB writes so that the writes return the AHB bus ready when the AHB command is granted by an arbitrator and will not wait for the AHB command to complete. The use of this feature improves the write performance.

  The FlexSPI AHB TX buffer has 64 bytes, and the internal AHB can implement a write access burst with a maximum size of 32 bytes. Therefore, the result of the write access performance is higher when transferring 32 bytes.

  In the *flexspi_hyper_ram_polling_transfer.c* file:

```
config.ahbConfig.enableAHBBufferable = true;
```

- Enable the FlexSPI read access prefetch feature:

  When the FlexSPI AHB read prefetch is enabled, the FlexSPI will fetch more flash/RAM read data than what is needed for the current AHB burst. This reduces the latency for the next AHB read access, improving the read access performance.

  In the *flexspi_hyper_ram_polling_transfer.c* file:

```
config.ahbConfig.enableAHBPrefetch = true;
```

  Even with the FlexSPI read prefetch enabled, the read access performance is not as good as it could be. The key reason is the invalidation of the internal AHB read burst. The disabled Dcache will limit the AHB read access in the single mode. When the Dcache is enabled, the AHB read access can be implemented in the INCR burst mode and the performance improves, as shown in Table 5 and Figure 16.

In the *flexspi_hyper_ram_polling_transfer.c* file:

```
/* SCB_DisableDCache(); */
```

Table 5. HyperRAM core read performance with Dcache enabled

—	Byte	4	8	32	128	512	1024	16 K	32 K
Read (Disable DCache)	Time (ns)	278	338	696	2140	7896	15576	248876	497730
	Perf (MB/S)	14	23	45	59	64	65	65	65
Read (Enable DCache)	Time (ns)	275	285	351	671	1955	3661	58248	116501
	Perf (MB/S)	14	28	91	190	261	279	281	281

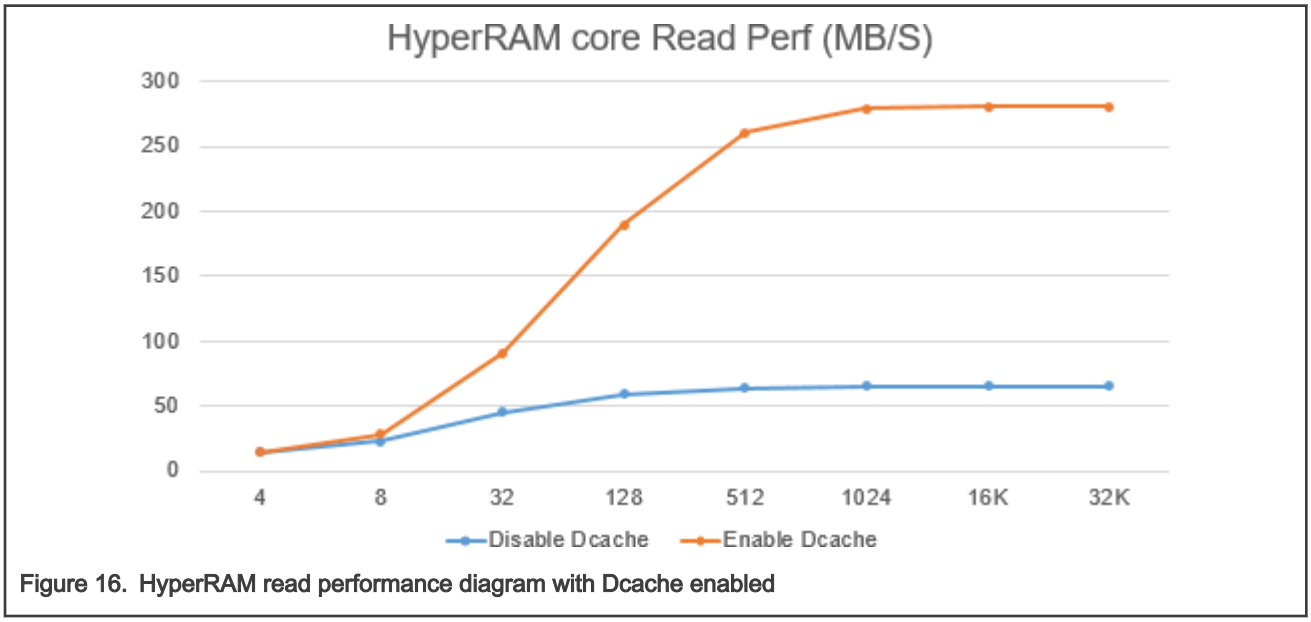

Figure 16. HyperRAM read performance diagram with Dcache enabled

6 Validated HyperRAM devices

HyperRAM devices from different vendors are tested to validate if they can work well with i.MX RT series.

Table 6 lists the test results of all HyperRAM devices. As shown in Table 6, some HyperRAM device, such as **7KS0641DPHI02**, cannot pass the test. Therefore, when using HyperRAM on i.MX RT series, all passed devices in Table 6 are recommended to use.

Table 6. HyperRam device test results

RT part number	HyperRAM vendor	HyperRAM part number	Results
PIMXRT1176DVMAA	Cypress	7KL0642DPHB02 (8 MB)	PASS
PIMXRT1064DVL6A	Cypress	7KS0642GAHI02 (8 MB)	PASS
PIMXRT1052DVL6B	Cypress	7KS0641DPHI02 (8 MB)	FAIL
PIMXRT1064DVL6A	Cypress	7KS0641DPHI02 (8 MB)	FAIL
PIMXRT1064DVL6A	Winbond	W956x8MBYA (8 MB)	PASS
PIMXRT1064DVL6A	ISSI	IS66WVH32M8DALL (32 MB)	FAIL
PIMXRT1176DVMAA	ISSI	IS66WVH8M8DALL (8 MB)	FAIL

7 Conclusion

This application note describes how to enable the HyperRAM device with the i.MX RT1050 FlexSPI interface, provides the example source code for a quick reference, and also analyzes the performance of the HyperRAM access according to the test results. For more details, see the following:

- *i.MX RT1050 Reference Manual* (document IMXRT1050RM)
- HyperBus TM Specification Low Signal Count High Performance DDR Bus
- S27KS0641 user manual from Cypress

8 Revision history

Table 7. Revision history

Revision number	Date	Substantive changes
0	August 2018	Initial release
1	August 2020	Added Validated HyperRAM devices
2	7 May, 2021	Updated Validated HyperRAM devices

How To Reach Us

Home Page:

nxp.com

Web Support:

nxp.com/support

For more information, please visit: http://www.nxp.com

For sales office addresses, please send an email to: salesaddresses@nxp.com

Date of release: 7 May, 2021

Document identifier: AN12239

arm