



Chipsmall Limited consists of a professional team with an average of over 10 year of expertise in the distribution of electronic components. Based in Hongkong, we have already established firm and mutual-benefit business relationships with customers from,Europe,America and south Asia,supplying obsolete and hard-to-find components to meet their specific needs.

With the principle of “Quality Parts,Customers Priority,Honest Operation,and Considerate Service”,our business mainly focus on the distribution of electronic components. Line cards we deal with include Microchip,ALPS,ROHM,Xilinx,Pulse,ON,Everlight and Freescale. Main products comprise IC,Modules,Potentiometer,IC Socket,Relay,Connector.Our parts cover such applications as commercial,industrial, and automotives areas.

We are looking forward to setting up business relationship with you and hope to provide you with the best service and solution. Let us make a better world for our industry!



Contact us

Tel: +86-755-8981 8866 Fax: +86-755-8427 6832

Email & Skype: info@chipsmall.com Web: www.chipsmall.com

Address: A1208, Overseas Decoration Building, #122 Zhenhua RD., Futian, Shenzhen, China





This datasheet describes serial configuration (EPCS) devices.

Supported Devices

Table 1 lists the supported Altera® EPCS devices.

Table 1. Altera EPCS Devices

Device	Memory Size (bits)	On-Chip Decompression Support	ISP Support	Cascading Support	Reprogrammable	Recommended Operating Voltage (V)
EPCS1	1,048,576	No	Yes	No	Yes	3.3
EPCS4	4,194,304	No	Yes	No	Yes	3.3
EPCS16	16,777,216	No	Yes	No	Yes	3.3
EPCS64	67,108,864	No	Yes	No	Yes	3.3
EPCS128	134,217,728	No	Yes	No	Yes	3.3

- 
 For more information about programming EPCS devices using the Altera Programming Unit (APU) or Master Programming Unit (MPU), refer to the [Altera Programming Hardware Datasheet](#).
- 
 The EPCS device can be re-programmed in system with ByteBlaster™ II download cable or an external microprocessor using SRunner. For more information, refer to [AN418: SRunner: An Embedded Solution for Serial Configuration Device Programming](#).

Features

EPCS devices offer the following features:

- Supports active serial (AS) x1 configuration scheme
- Easy-to-use four-pin interface
- Low cost, low pin count, and non-volatile memory
- Low current during configuration and near-zero standby mode current
- 2.7-V to 3.6-V operation
- EPCS1, EPCS4, and EPCS16 devices available in 8-pin small-outline integrated circuit (SOIC) package
- EPCS64 and EPCS128 devices available in 16-pin SOIC package

- Enables the Nios® processor to access unused flash memory through AS memory interface
- Reprogrammable memory with more than 100,000 erase or program cycles
- Write protection support for memory sectors using status register bits
- In-system programming (ISP) support with SRunner software driver
- ISP support with USB-Blaster™, EthernetBlaster, or ByteBlaster II download cables
- Additional programming support with the APU and programming hardware from BP Microsystems, System General, and other vendors
- By default, the memory array is erased and the bits are set to 1

Functional Description

To configure a system using an SRAM-based device, each time you power on the device, you must load the configuration data. The EPCS device is a flash memory device that can store configuration data that you use for FPGA configuration purpose after power on. You can use the EPCS device on all FPGA that support AS x1 configuration scheme.

For an 8-pin SOIC package, you can migrate vertically from the EPCS1 device to the EPCS4 or EPCS16 device. For a 16-pin SOIC package, you can migrate vertically from the EPCS64 device to the EPCS128 device.

With the new data decompression feature supported, you can determine using which EPCS device to store the configuration data for configuring your FPGA.

Example 1 shows how you can calculate the compression ratio to determine which EPCS device is suitable for the FPGA.

Example 1. Compression Ratio Calculation

EP4SGX530 = 189,000,000 bits

EPCS128 = 134,217,728 bits

Preliminary data indicates that compression typically reduces the configuration bitstream size by 35% to 55%. Assume worst case that is 35% decompression.

$189,000,000 \text{ bits} \times 0.65 = 122,850,000 \text{ bits}$

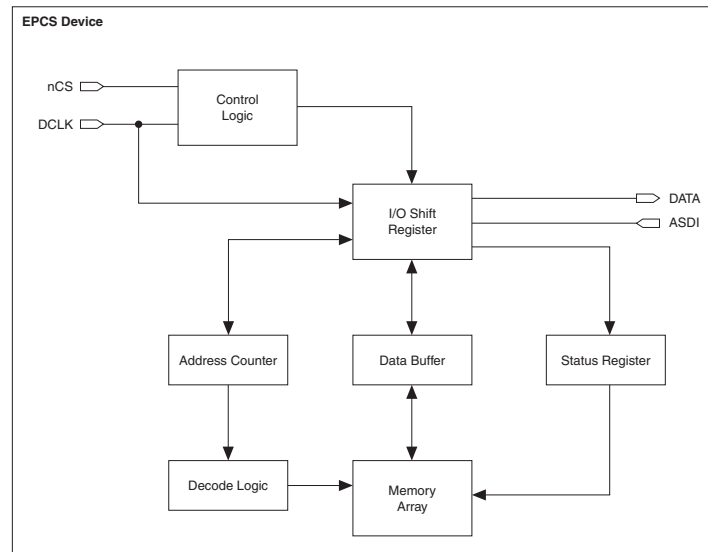
The EPCS128 device is suitable.



For more information about the FPGA decompression feature, refer to the configuration chapter in the appropriate device handbook.

Figure 1 shows the EPCS device block diagram.

Figure 1. EPCS Device Block Diagram



Accessing Memory in EPCS Devices

You can access the unused memory locations of the EPCS device to store or retrieve data through the Nios processor and SOPC Builder. SOPC Builder is an Altera tool for creating bus-based (especially microprocessor-based) systems in Altera devices. SOPC Builder assembles library components such as processors and memories into custom microprocessor systems.

SOPC Builder includes the EPCS device controller core, which is an interface core designed specifically to work with the EPCS device. With this core, you can create a system with a Nios embedded processor that allows software access to any memory location within the EPCS device.

Active Serial FPGA Configuration

The following Altera FPGAs support the AS configuration scheme with EPCS devices:

- Arria® series
- Cyclone® series
- All device families in the Stratix® series except the Stratix device family

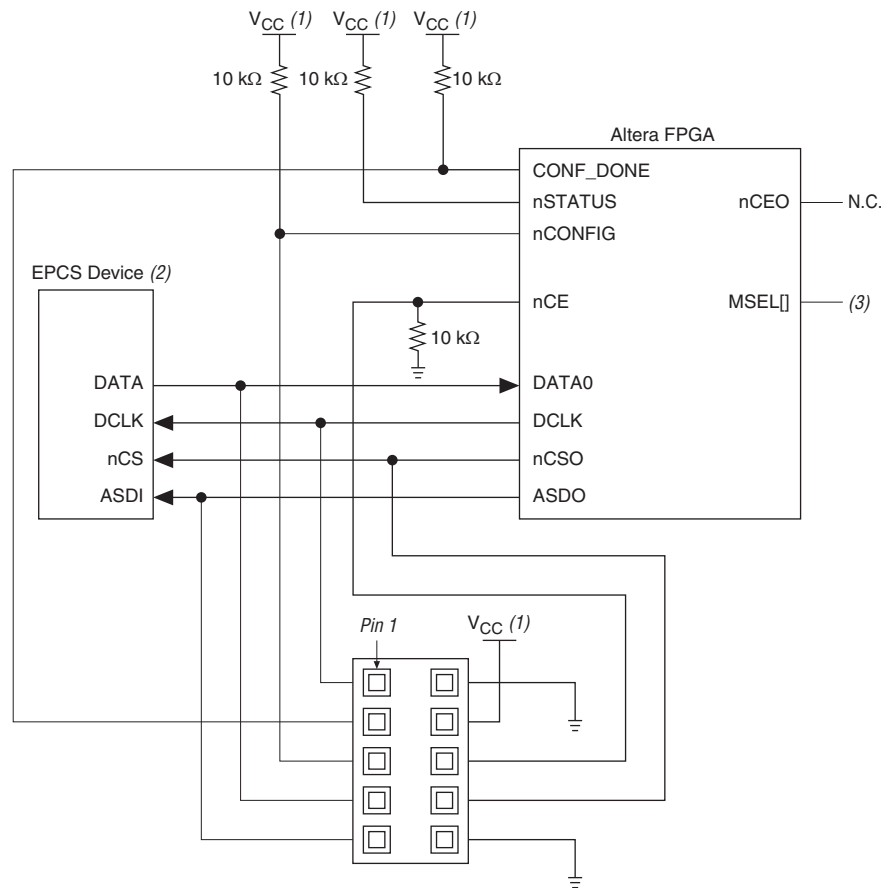
There are four signals on the EPCS device that interface directly with the FPGA's control signals. The EPCS device signals are **DATA**, **DCLK**, **ASDI**, and **nCS** interface with the **DATA0**, **DCLK**, **ASDO**, and **nCS0** control signals on the FPGA, respectively.



For more information about the EPCS device pin description, refer to [Table 23 on page 36](#).

Figure 2 shows the configuration of an FPGA device in the AS configuration scheme with an EPCS device using a download cable.

Figure 2. Altera FPGA Configuration in AS Mode Using a Download Cable (1), (4)

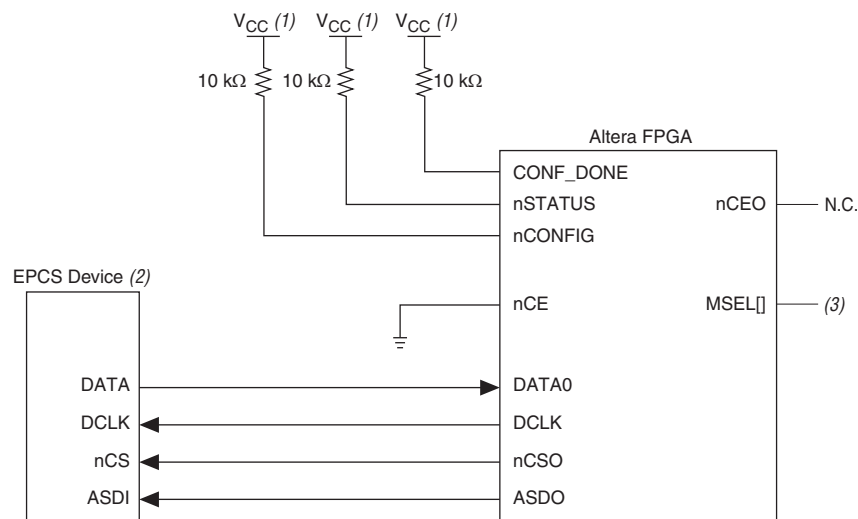


Notes to Figure 2:

- (1) For more information about the V_{CC} value, refer to the configuration chapter in the appropriate device handbook.
- (2) EPCS devices cannot be cascaded.
- (3) Connect the $MSEL[]$ input pins to select the AS configuration mode. For more information, refer to the configuration chapter in the appropriate device handbook.
- (4) For more information about configuration pin I/O requirements in an AS configuration scheme for an Altera FPGA, refer to the configuration chapter in the appropriate device handbook.

Figure 3 shows the configuration of an FPGA device in the AS configuration scheme with an EPCS device using the APU or a third-party programmer.


Figure 3. Altera FPGA Configuration in AS Mode Using APU or a Third-party Programmer (1), (4)




Notes to Figure 3:

- (1) For more information about the V_{CC} value, refer to the configuration chapter in the appropriate device handbook.
- (2) EPCS devices cannot be cascaded.
- (3) Connect the $MSEL[]$ input pins to select the AS configuration mode. For more information, refer to the configuration chapter in the appropriate device handbook.
- (4) For more information about configuration pin I/O requirements in an AS configuration scheme for an Altera FPGA, refer to the configuration chapter in the appropriate device handbook.

In an AS configuration, the FPGA acts as the configuration master in the configuration flow and provides the clock to the EPCS device. The FPGA enables the EPCS device by pulling the nCS signal low using the $nCSO$ signal as shown in Figure 2 and Figure 3. Then, the FPGA sends the instructions and addresses to the EPCS device using the $ASDO$ signal. The EPCS device responds to the instructions by sending the configuration data to the FPGA's $DATA0$ pin on the falling edge of $DCLK$. The data is latched into the FPGA on the next $DCLK$ signal's falling edge.

 Before the FPGA enters configuration mode, ensure that V_{CC} of the EPCS device is ready. If V_{CC} is not ready, you must hold $nCONFIG$ low until all power rails of EPCS device are ready.

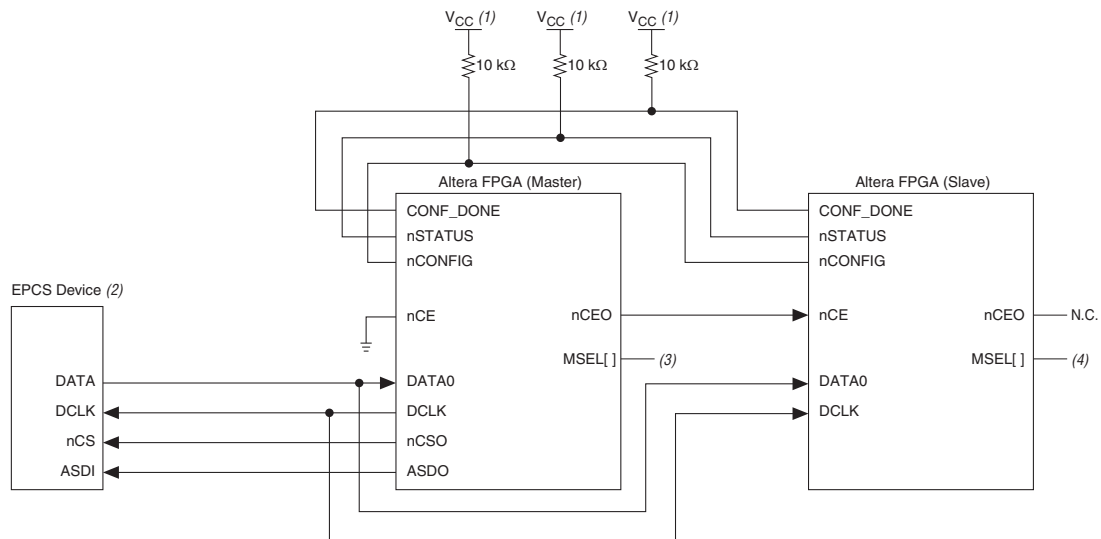
The FPGA controls the $nSTATUS$ and $CONF_DONE$ pins during configuration in the AS mode. If the $CONF_DONE$ signal does not go high at the end of configuration, or if the signal goes high too early, the FPGA pulses its $nSTATUS$ pin low to start a reconfiguration. If the configuration is successful, the FPGA releases the $CONF_DONE$ pin, allowing the external 10-k Ω resistor to pull the $CONF_DONE$ signal high. The FPGA initialization begins after the $CONF_DONE$ pin goes high. After the initialization, the FPGA enters user mode.

 For more information about configuring the FPGAs in AS configuration mode or other configuration modes, refer to the configuration chapter in the appropriate device handbook.

You can configure multiple devices with a single EPCS device. However, you cannot cascade EPCS devices. To ensure that the programming file size of the cascaded FPGAs does not exceed the capacity of an EPCS device, refer to [Table 1 on page 1](#).

[Figure 4](#) shows the AS configuration scheme with multiple FPGAs in the chain. The first FPGA is the configuration master and its $MSEL[]$ pins are set to AS mode. The following FPGAs are configuration slave devices and their $MSEL[]$ pins are set to PS mode.

Figure 4. Multiple Devices in AS Mode (1), (5)



Notes to Figure 4:

- (1) For more information about the V_{CC} value, refer to the configuration chapter in the appropriate device handbook.
- (2) EPCS devices cannot be cascaded.
- (3) Connect the $MSEL[]$ input pins to select the AS configuration mode. For more information, refer to the configuration chapter in the appropriate device handbook.
- (4) Connect the $MSEL[]$ input pins to select the PS configuration mode. For more information, refer to the configuration chapter in the appropriate device handbook.
- (5) For more information about configuration pin I/O requirements in an AS configuration scheme for an Altera FPGA, refer to the configuration chapter in the appropriate device handbook.

EPCS Device Memory Access

This section describes the memory array organization and operation codes of the EPCS device. For the timing specifications, refer to [“Timing Information” on page 29](#).

Memory Array Organization

[Table 2](#) lists the memory array organization details in EPCS128, EPCS64, EPCS16, EPCS4, and EPCS1 devices.

Table 2. Memory Array Organization in EPCS Devices

Details	EPCS128	EPCS64	EPCS16	EPCS4	EPCS1
Bytes	16,777,216 bytes (128 Mb)	8,388,608 bytes (64 Mb)	2,097,152 bytes (16 Mb)	524,288 bytes (4 Mb)	131,072 bytes (1 Mb)
Number of sectors	64	128	32	8	4
Bytes per sector	262,144 bytes (2 Mb)	65,536 bytes (512 Kb)	65,536 bytes (512 Kb)	65,536 bytes (512 Kb)	32,768 bytes (256 Kb)
Pages per sector	1,024	256	256	256	128
Total number of pages	65,536	32,768	8,192	2,048	512
Bytes per page	256 bytes	256 bytes	256 bytes	256 bytes	256 bytes

[Table 3](#) through [Table 7](#) on [page 12](#) list the address range for each sector in EPCS1, EPCS4, EPCS16, EPCS64, and EPCS128 devices.

Table 3. Address Range for Sectors in EPCS1 Devices

Sector	Address Range (byte Addresses in HEX)	
	Start	End
3	H'18000	H'1FFFF
2	H'10000	H'17FFF
1	H'08000	H'0FFFF
0	H'00000	H'07FFF

Table 4. Address Range for Sectors in EPCS4 Devices

Sector	Address Range (Byte Addresses in HEX)	
	Start	End
7	H'70000	H'7FFFF
6	H'60000	H'6FFFF
5	H'50000	H'5FFFF
4	H'40000	H'4FFFF
3	H'30000	H'3FFFF
2	H'20000	H'2FFFF
1	H'10000	H'1FFFF
0	H'00000	H'0FFFF

Table 5. Address Range for Sectors in EPCS16 Devices

Sector	Address Range (Byte Addresses in HEX)	
	Start	End
31	H'1F0000	H'1FFFFFF
30	H'1E0000	H'1EFFFFFF
29	H'1D0000	H'1DFFFFFF
28	H'1C0000	H'1CFFFFFF
27	H'1B0000	H'1BFFFFFF
26	H'1A0000	H'1AFFFFFF
25	H'190000	H'19FFFFFF
24	H'180000	H'18FFFFFF
23	H'170000	H'17FFFFFF
22	H'160000	H'16FFFFFF
21	H'150000	H'15FFFFFF
20	H'140000	H'14FFFFFF
19	H'130000	H'13FFFFFF
18	H'120000	H'12FFFFFF
17	H'110000	H'11FFFFFF
16	H'100000	H'10FFFFFF
15	H'0F0000	H'0FFFFFF
14	H'0E0000	H'0EFFFFFF
13	H'0D0000	H'0DFFFFFF
12	H'0C0000	H'0CFFFFFF
11	H'0B0000	H'0BFFFFFF
10	H'0A0000	H'0AFFFFFF
9	H'090000	H'09FFFFFF
8	H'080000	H'08FFFFFF
7	H'070000	H'07FFFFFF
6	H'060000	H'06FFFFFF
5	H'050000	H'05FFFFFF
4	H'040000	H'04FFFFFF
3	H'030000	H'03FFFFFF
2	H'020000	H'02FFFFFF
1	H'010000	H'01FFFFFF
0	H'000000	H'00FFFFFF

Table 6. Address Range for Sectors in EPCS64 Devices (Part 1 of 4)

Sector	Address Range (Byte Addresses in HEX)	
	Start	End
127	H'7F0000	H'7FFFFFFF
126	H'7E0000	H'7EFFFFFF
125	H'7D0000	H'7DFFFFFF
124	H'7C0000	H'7CFFFFFF
123	H'7B0000	H'7BFFFFFF
122	H'7A0000	H'7AFFFFFF
121	H'790000	H'79FFFFFF
120	H'780000	H'78FFFFFF
119	H'770000	H'77FFFFFF
118	H'760000	H'76FFFFFF
117	H'750000	H'75FFFFFF
116	H'740000	H'74FFFFFF
115	H'730000	H'73FFFFFF
114	H'720000	H'72FFFFFF
113	H'710000	H'71FFFFFF
112	H'700000	H'70FFFFFF
111	H'6F0000	H'6FFFFFFF
110	H'6E0000	H'6EFFFFFF
109	H'6D0000	H'6DFFFFFF
108	H'6C0000	H'6CFFFFFF
107	H'6B0000	H'6BFFFFFF
106	H'6A0000	H'6AFFFFFF
105	H'690000	H'69FFFFFF
104	H'680000	H'68FFFFFF
103	H'670000	H'67FFFFFF
102	H'660000	H'66FFFFFF
101	H'650000	H'65FFFFFF
100	H'640000	H'64FFFFFF
99	H'630000	H'63FFFFFF
98	H'620000	H'62FFFFFF
97	H'610000	H'61FFFFFF
96	H'600000	H'60FFFFFF
95	H'5F0000	H'5FFFFFFF
94	H'5E0000	H'5EFFFFFF
93	H'5D0000	H'5DFFFFFF
92	H'5C0000	H'5CFFFFFF
91	H'5B0000	H'5BFFFFFF
90	H'5A0000	H'5AFFFFFF

Table 6. Address Range for Sectors in EPCS64 Devices (Part 2 of 4)

Sector	Address Range (Byte Addresses in HEX)	
	Start	End
89	H'590000	H'59FFFF
88	H'580000	H'58FFFF
87	H'570000	H'57FFFF
86	H'560000	H'56FFFF
85	H'550000	H'55FFFF
84	H'540000	H'54FFFF
83	H'530000	H'53FFFF
82	H'520000	H'52FFFF
81	H'510000	H'51FFFF
80	H'500000	H'50FFFF
79	H'4F0000	H'4FFFFF
78	H'4E0000	H'4EFFFF
77	H'4D0000	H'4DFFFF
76	H'4C0000	H'4CFFFF
75	H'4B0000	H'4BFFFF
74	H'4A0000	H'4AFFFF
73	H'490000	H'49FFFF
72	H'480000	H'48FFFF
71	H'470000	H'47FFFF
70	H'460000	H'46FFFF
69	H'450000	H'45FFFF
68	H'440000	H'44FFFF
67	H'430000	H'43FFFF
66	H'420000	H'42FFFF
65	H'410000	H'41FFFF
64	H'400000	H'40FFFF
63	H'3F0000	H'3FFFFF
62	H'3E0000	H'3EFFFF
61	H'3D0000	H'3DFFFF
60	H'3C0000	H'3CFFFF
59	H'3B0000	H'3BFFFF
58	H'3A0000	H'3AFFFF
57	H'390000	H'39FFFF
56	H'380000	H'38FFFF
55	H'370000	H'37FFFF
54	H'360000	H'36FFFF
53	H'350000	H'35FFFF
52	H'340000	H'34FFFF

Table 6. Address Range for Sectors in EPCS64 Devices (Part 3 of 4)

Sector	Address Range (Byte Addresses in HEX)	
	Start	End
51	H'330000	H'33FFFF
50	H'320000	H'32FFFF
49	H'310000	H'31FFFF
48	H'300000	H'30FFFF
47	H'2F0000	H'2FFFFFFF
46	H'2E0000	H'2EFFFF
45	H'2D0000	H'2DFFFF
44	H'2C0000	H'2CFFFF
43	H'2B0000	H'2BFFFF
42	H'2A0000	H'2AFFFF
41	H'290000	H'29FFFF
40	H'280000	H'28FFFF
39	H'270000	H'27FFFF
38	H'260000	H'26FFFF
37	H'250000	H'25FFFF
36	H'240000	H'24FFFF
35	H'230000	H'23FFFF
34	H'220000	H'22FFFF
33	H'210000	H'21FFFF
32	H'200000	H'20FFFF
31	H'1F0000	H'1FFFFFFF
30	H'1E0000	H'1EFFFF
29	H'1D0000	H'1DFFFF
28	H'1C0000	H'1CFFFF
27	H'1B0000	H'1BFFFF
26	H'1A0000	H'1AFFFF
25	H'190000	H'19FFFF
24	H'180000	H'18FFFF
23	H'170000	H'17FFFF
22	H'160000	H'16FFFF
21	H'150000	H'15FFFF
20	H'140000	H'14FFFF
19	H'130000	H'13FFFF
18	H'120000	H'12FFFF
17	H'110000	H'11FFFF
16	H'100000	H'10FFFF
15	H'0F0000	H'0FFFFFFF
14	H'0E0000	H'0EFFFF

Table 6. Address Range for Sectors in EPCS64 Devices (Part 4 of 4)

Sector	Address Range (Byte Addresses in HEX)	
	Start	End
13	H'0D0000	H'0DFFFF
12	H'0C0000	H'0CFFFF
11	H'0B0000	H'0BFFFF
10	H'0A0000	H'0AFFFF
9	H'090000	H'09FFFF
8	H'080000	H'08FFFF
7	H'070000	H'07FFFF
6	H'060000	H'06FFFF
5	H'050000	H'05FFFF
4	H'040000	H'04FFFF
3	H'030000	H'03FFFF
2	H'020000	H'02FFFF
1	H'010000	H'01FFFF
0	H'000000	H'00FFFF

Table 7. Address Range for Sectors in EPCS128 Devices (Part 1 of 3)

Sector	Address Range (Byte Addresses in HEX)	
	Start	End
63	H'FC0000	H'FFFFFF
62	H'F80000	H'FBFFFF
61	H'F40000	H'F7FFFF
60	H'F00000	H'F3FFFF
59	H'EC0000	H'EFFFFF
58	H'E80000	H'EBFFFF
57	H'E40000	H'E7FFFF
56	H'E00000	H'E3FFFF
55	H'DC0000	H'DFffff
54	H'D80000	H'DBFFFF
53	H'D40000	H'D7FFFF
52	H'D00000	H'D3FFFF
51	H'CC0000	H'CFffff
50	H'C80000	H'CBFFFF
49	H'C40000	H'C7FFFF
48	H'C00000	H'C3FFFF
47	H'BC0000	H'BFffff
46	H'B80000	H'BBFFFF
45	H'B40000	H'B7FFFF
44	H'B00000	H'B3FFFF

Table 7. Address Range for Sectors in EPCS128 Devices (Part 2 of 3)

Sector	Address Range (Byte Addresses in HEX)	
	Start	End
43	H'AC0000	H'AFFFFF
42	H'A80000	H'ABFFFF
41	H'A40000	H'A7FFFF
40	H'A00000	H'A3FFFF
39	H'9C0000	H'9FFFFF
38	H'980000	H'9BFFFF
37	H'940000	H'97FFFF
36	H'900000	H'93FFFF
35	H'8C0000	H'8FFFFF
34	H'880000	H'8BFFFF
33	H'840000	H'87FFFF
32	H'800000	H'83FFFF
31	H'7C0000	H'7FFFFF
30	H'780000	H'7BFFFF
29	H'740000	H'77FFFF
28	H'700000	H'73FFFF
27	H'6C0000	H'6FFFFF
26	H'680000	H'6BFFFF
25	H'640000	H'67FFFF
24	H'600000	H'63FFFF
23	H'5C0000	H'5FFFFF
22	H'580000	H'5BFFFF
21	H'540000	H'57FFFF
20	H'500000	H'53FFFF
19	H'4C0000	H'4FFFFF
18	H'480000	H'4BFFFF
17	H'440000	H'47FFFF
16	H'400000	H'43FFFF
15	H'3C0000	H'3FFFFF
14	H'380000	H'3BFFFF
13	H'340000	H'37FFFF
12	H'300000	H'33FFFF
11	H'2C0000	H'2FFFFF
10	H'280000	H'2BFFFF
9	H'240000	H'27FFFF
8	H'200000	H'23FFFF
7	H'1C0000	H'1FFFFF
6	H'180000	H'1BFFFF

Table 7. Address Range for Sectors in EPCS128 Devices (Part 3 of 3)

Sector	Address Range (Byte Addresses in HEX)	
	Start	End
5	H'140000	H'17FFFF
4	H'100000	H'13FFFF
3	H'0C0000	H'0FFFFFF
2	H'080000	H'0BFFFF
1	H'040000	H'07FFFF
0	H'000000	H'03FFFF

Operation Codes

This section describes the operations that you can use to access the memory in EPCS devices. Use the DATA, DCLK, ASDI, and nCS signals to access the memory in EPCS devices. When performing the operation, addresses and data are shifted in and out of the device serially, with MSB first.

The device samples the AS data input on the first rising edge of the DCLK after the active low chip select (nCS) input signal is driven low. Shift the operation code, with MSB first, into the EPCS device serially through the AS data input (ASDI) pin. Each operation code bit is latched into the EPCS device on the rising edge of the DCLK.

Different operations require a different sequence of inputs. While executing an operation, you must shift in the desired operation code, followed by the address bytes or data bytes, both address and data bytes, or none of them. The device must drive nCS pin high after the last bit of the operation sequence is shifted in. Table 8 lists the operation sequence for every operation supported by the EPCS devices.

For read operations, the data read is shifted out on the DATA pin. You can drive the nCS pin high after any bit of the data-out sequence is shifted out.

For write and erase operations, drive the nCS pin high at a byte boundary that is in a multiple of eight clock pulses. Otherwise, the operation is rejected and not executed.

All attempts to access the memory contents while a write or erase cycle is in progress are rejected, and the write or erase cycle will continue unaffected.

Table 8. EPCS Devices Operation Codes

Operation	Operation Code ⁽¹⁾	Address Bytes	Dummy Bytes	Data Bytes	DCLK f _{MAX} (MHz)
Write enable	0000 0110	0	0	0	25
Write disable	0000 0100	0	0	0	25
Read status	0000 0101	0	0	1 to infinite ⁽²⁾	32
Read bytes	0000 0011	3	0	1 to infinite ⁽²⁾	20
Read silicon ID ⁽⁴⁾	1010 1011	0	3	1 to infinite ⁽²⁾	32
Fast read	0000 1011	3	1	1 to infinite ⁽²⁾	40
Write status	0000 0001	0	0	1	25
Write bytes	0000 0010	3	0	1 to 256 ⁽³⁾	25
Erase bulk	1100 0111	0	0	0	25

Table 8. EPCS Devices Operation Codes

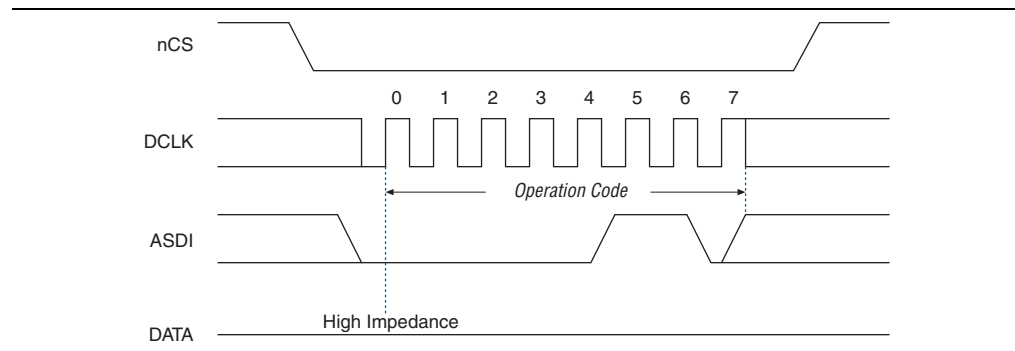
Operation	Operation Code ⁽¹⁾	Address Bytes	Dummy Bytes	Data Bytes	DCLK f _{MAX} (MHz)
Erase sector	1101 1000	3	0	0	25
Read device identification ⁽⁵⁾	1001 1111	0	2	1 to infinite ⁽²⁾	25

Notes to Table 8:

- (1) List MSB first and LSB last.
- (2) The status register, data, or silicon ID is read out at least once on the DATA pin and is continuously read out until the nCS pin is driven high.
- (3) A write bytes operation requires at least one data byte on the DATA pin. If more than 256 bytes are sent to the device, only the last 256 bytes are written to the memory.
- (4) The read silicon ID operation is available only for EPCS1, EPCS4, EPCS16, and EPCS64 devices.
- (5) The read device identification operation is available only for EPCS128 devices.

Write Enable Operation

The write enable operation code is b'0000 0110, and it lists the MSB first. The write enable operation sets the write enable latch bit, which is bit 1 in the status register. Always set the write enable latch bit before write bytes, write status, erase bulk, and erase sector operations. Figure 5 shows the instruction sequence of the write enable operation.

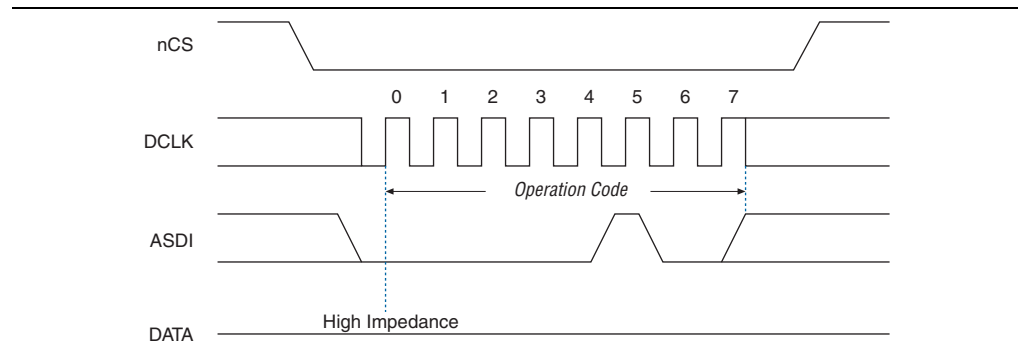
Figure 5. Write Enable Operation Timing Diagram**Write Disable Operation**

The write disable operation code is b'0000 0100 and it lists the MSB first. The write disable operation resets the write enable latch bit, which is bit 1 in the status register. To prevent the memory from being written unintentionally, the write enable latch bit is automatically reset when implementing the write disable operation, and under the following conditions:

- Power up
- Write bytes operation completion
- Write status operation completion
- Erase bulk operation completion
- Erase sector operation completion

Figure 6 shows the instruction sequence of the write disable operation.

Figure 6. Write Disable Operation Timing Diagram



Read Status Operation

The read status operation code is $b'0000\ 0101$ and it lists the MSB first. You can use the read status operation to read the status register. Figure 7 and Figure 8 show the status bits in the status register of the EPCS devices.

Figure 7. EPCS128, EPCS64, EPCS16, and EPCS4 Status Register Status Bits

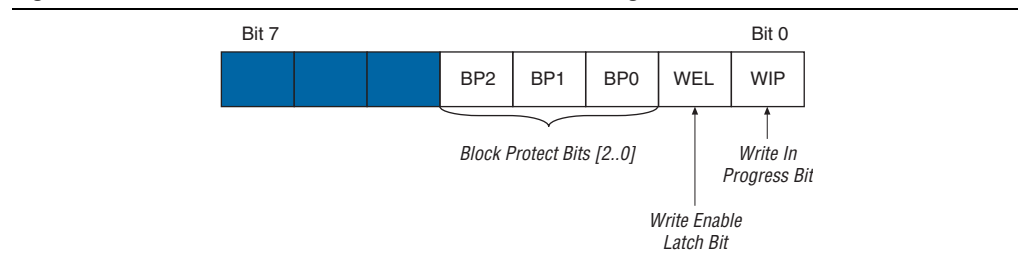
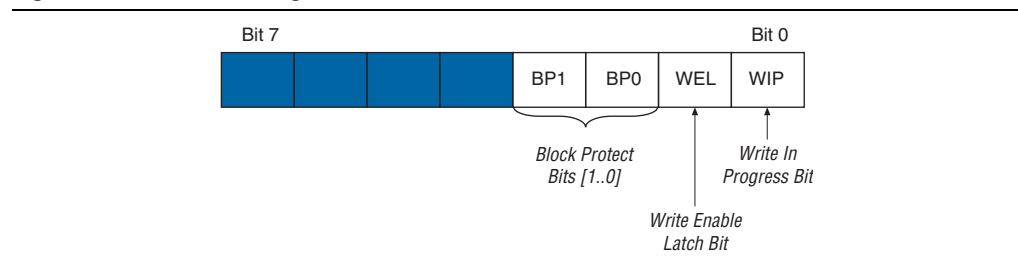


Figure 8. EPCS1 Status Register Status Bits



Setting the write in progress bit to 1 indicates that the EPCS device is busy with a write or erase cycle. Resetting the write in progress bit to 0 indicates no write or erase cycle is in progress.

Resetting the write enable latch bit to 0 indicates that no write or erase cycle is accepted. Set the write enable latch bit to 1 before every write bytes, write status, erase bulk, and erase sector operations.

The non-volatile block protect bits determine the area of the memory protected from being written or erased unintentionally. [Table 9](#) through [Table 13 on page 19](#) list the protected area in the EPCS devices with reference to the block protect bits. The erase bulk operation is only available when all the block protect bits are set to 0. When any of the block protect bits are set to 1, the relevant area is protected from being written by a write bytes operation or erased by an erase sector operation.

Table 9. Block Protection Bits in the EPCS1 Device

Status Register Content		Memory Content	
BP1 Bit	BPO Bit	Protected Area	Unprotected Area
0	0	None	All four sectors—0 to 3
0	1	Sector 3	Three sectors—0 to 2
1	0	Two sectors—2 and 3	Two sectors—0 and 1
1	1	All sectors	None

Table 10. Block Protection Bits in the EPCS4 Device

Status Register Content			Memory Content	
BP2 Bit	BP1 Bit	BPO Bit	Protected Area	Unprotected Area
0	0	0	None	All eight sectors—0 to 7
0	0	1	Sector 7	Seven sectors—0 to 6
0	1	0	Sectors 6 and 7	Six sectors—0 to 5
0	1	1	Four sectors—4 to 7	Four sectors—0 to 3
1	0	0	All sectors	None
1	0	1	All sectors	None
1	1	0	All sectors	None
1	1	1	All sectors	None

Table 11. Block Protection Bits in the EPCS16 Device

Status Register Content			Memory Content	
BP2 Bit	BP1 Bit	BPO Bit	Protected Area	Unprotected Area
0	0	0	None	All sectors (32 sectors 0 to 31)
0	0	1	Upper 32nd (Sector 31)	Lower 31/32nds (31 sectors—0 to 30)
0	1	0	Upper sixteenth (two sectors—30 and 31)	Lower 15/16ths (30 sectors—0 to 29)
0	1	1	Upper eighth (four sectors—28 to 31)	Lower seven-eighths (28 sectors—0 to 27)
1	0	0	Upper quarter (eight sectors—24 to 31)	Lower three-quarters (24 sectors—0 to 23)
1	0	1	Upper half (sixteen sectors—16 to 31)	Lower half (16 sectors—0 to 15)
1	1	0	All sectors (32 sectors—0 to 31)	None
1	1	1	All sectors (32 sectors—0 to 31)	None

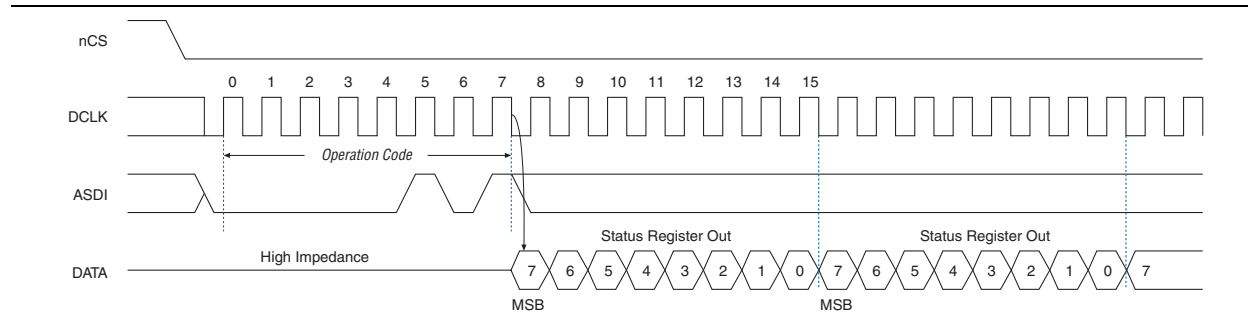
Table 12. Block Protection Bits in the EPCS64 Devices

Status Register Content			Memory Content	
BP2 Bit	BP1 Bit	BPO Bit	Protected Area	Unprotected Area
0	0	0	None	All sectors (128 sectors: 0 to 127)
0	0	1	Upper 64th (2 sectors: 126 and 127)	Lower 63/64ths (126 sectors: 0 to 125)
0	1	0	Upper 32nd (4 sectors: 124 to 127)	Lower 31/32nds (124 sectors: 0 to 123)
0	1	1	Upper sixteenth (8 sectors: 120 to 127)	Lower 15/16ths (120 sectors: 0 to 119)
1	0	0	Upper eighth (16 sectors: 112 to 127)	Lower seven-eighths (112 sectors: 0 to 111)
1	0	1	Upper quarter (32 sectors: 96 to 127)	Lower three-quarters (96 sectors: 0 to 95)
1	1	0	Upper half (64 sectors: 64 to 127)	Lower half (64 sectors: 0 to 63)
1	1	1	All sectors (128 sectors: 0 to 127)	None

Table 13. Block Protection Bits in the EPCS128 Device

Status Register Content			Memory Content	
BP2 Bit	BP1 Bit	BPO Bit	Protected Area	Unprotected Area
0	0	0	None	All sectors (64 sectors—0 to 63)
0	0	1	Upper 64th (1 sector—63)	Lower 63/64ths (63 sectors—0 to 62)
0	1	0	Upper 32nd (2 sectors—62 to 63)	Lower 31/32nds (62 sectors—0 to 61)
0	1	1	Upper 16th (4 sectors—60 to 63)	Lower 15/16ths (60 sectors—0 to 59)
1	0	0	Upper 8th (8 sectors—56 to 63)	Lower seven-eighths (56 sectors—0 to 55)
1	0	1	Upper quarter (16 sectors—48 to 63)	Lower three-quarters (48 sectors—0 to 47)
1	1	0	Upper half (32 sectors—32 to 63)	Lower half (32 sectors—0 to 31)
1	1	1	All sectors (64 sectors—0 to 63)	None

You can read the status register at any time, even during a write or erase cycle is in progress. When one of these cycles is in progress, you can check the write in progress bit (bit 0 of the status register) before sending a new operation to the device. The device can also read the status register continuously, as shown in [Figure 9](#).

Figure 9. Read Status Operation Timing Diagram

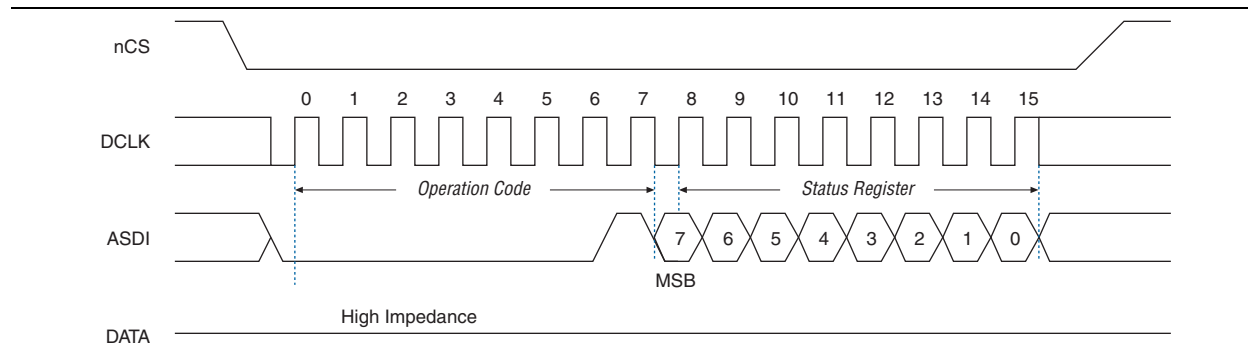
Write Status Operation

The write status operation code is $b'0000\ 0001$ and it lists the MSB first. Use the write status operation to set the status register block protection bits. The write status operation does not affect the other bits. Therefore, you can implement this operation to protect certain memory sectors, as listed in [Table 9](#) through [Table 13](#). After setting the block protect bits, the protected memory sectors are treated as read-only memory. You must execute the write enable operation before the write status operation so the device sets the status register's write enable latch bit to 1.

The write status operation is implemented by driving the nCS signal low, followed by shifting in the write status operation code and one data byte for the status register on the ASDI pin. [Figure 10](#) shows the instruction sequence of the write status operation. The nCS must be driven high after the eighth bit of the data byte has been latched in, otherwise the write status operation is not executed.

Immediately after the nCS signal drives high, the device initiates the self-timed write status cycle. The self-timed write status cycle usually takes 5 ms for all EPCS devices and is guaranteed to be less than 15 ms. For more information, refer to the t_{WS} value in [Table 16 on page 29](#). You must account for this delay to ensure that the status register is written with desired block protect bits. Alternatively, you can check the write in progress bit in the status register by executing the read status operation while the self-timed write status cycle is in progress. The write in progress bit is 1 during the self-timed write status cycle and 0 when it is complete.

Figure 10. Write Status Operation Timing Diagram



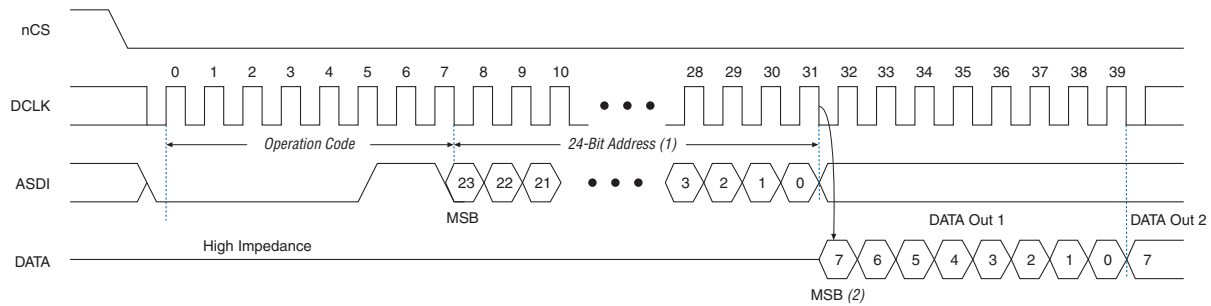
Read Bytes Operation

The read bytes operation code is $b'0000\ 0011$ and it lists the MSB first. To read the memory contents of the EPCS device, the device is first selected by driving the nCS signal low. Then, the read bytes operation code is shifted in followed by a 3-byte address ($A[23..0]$). Each address bit must be latched in on the rising edge of the DCLK signal. After the address is latched in, the memory contents of the specified address are shifted out serially on the DATA pin, beginning with the MSB. For reading Raw Programming Data files (.rpd), the content is shifted out serially beginning with the LSB. Each data bit is shifted out on the falling edge of the DCLK signal. The maximum DCLK frequency during the read bytes operation is 20 MHz.

The first byte address can be at any location. The device automatically increases the address to the next higher address after shifting out each byte of data. Therefore, the device can read the whole memory with a single read bytes operation. When the device reaches the highest address, the address counter restarts at 0x000000, allowing the memory contents to be read out indefinitely until the read bytes operation is terminated by driving the nCS signal high. The device can drive the nCS signal high at any time after data is shifted out. If the read bytes operation is shifted in while a write or erase cycle is in progress, the operation is not executed and does not affect the write or erase cycle in progress.

Figure 11 shows the instruction sequence of the read bytes operation.

Figure 11. Read Bytes Operation Timing Diagram



Notes to Figure 11:

- (1) Address bit A[23] is a don't-care bit in the EPCS64 device. Address bits A[23..21] are don't-care bits in the EPCS16 device. Address bits A[23..19] are don't-care bits in the EPCS4 device. Address bits A[23..17] are don't-care bits in the EPCS1 device.
- (2) For .rpd files, the read sequence shifts out the LSB of the data byte first.

Fast Read Operation

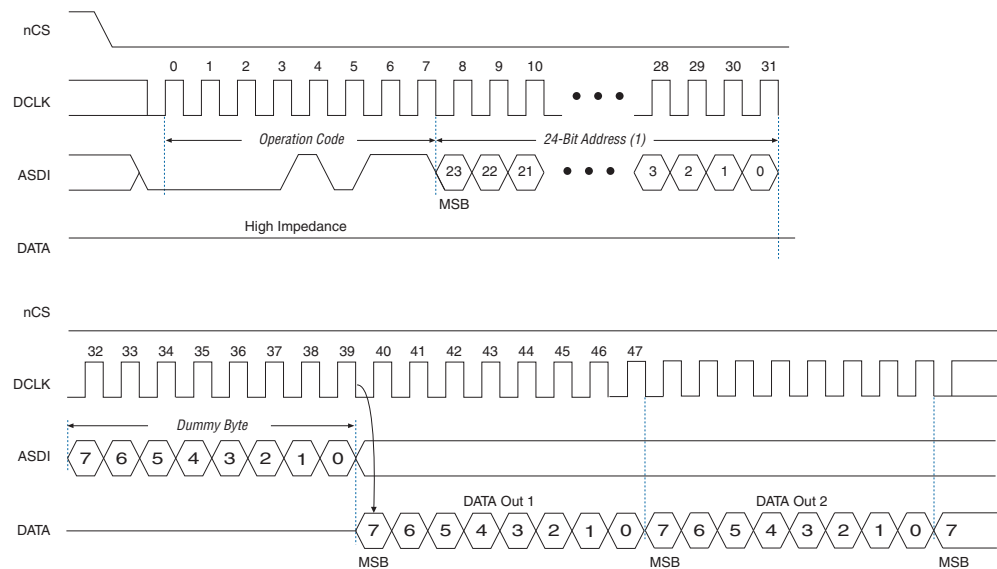
The fast read operation code is b'0000 1011 and it lists the MSB first. You can select the device by driving the nCS signal low. The fast read instruction code is followed by a 3-byte address (A23-A0) and a dummy byte with each bit being latched-in during the rising edge of the DCLK signal. Then, the memory contents at that address is shifted out on DATA with each bit being shifted out at a maximum frequency of 40 MHz during the falling edge of the DCLK signal.

The first addressed byte can be at any location. The address is automatically increased to the next higher address after each byte of data is shifted out. Therefore, the whole memory can be read with a single fast read instruction. When the highest address is reached, the address counter rolls over to 000000h, allowing the read sequence to continue indefinitely.

The fast read instruction is terminated by driving the nCS signal high at any time during data output. Any fast read instruction is rejected during the erase, program, or write operations without affecting the operation that is in progress.

Figure 12 shows the instruction sequence of the fast read operation.

Figure 12. Fast Read Operation Timing Diagram



Note to Figure 12:

- (1) Address bit A[23] is a don't-care bit in the EPCS64 device. Address bits A[23..21] are don't-care bits in the EPCS16 device. Address bits A[23..19] are don't-care bits in the EPCS4 device. Address bits A[23..17] are don't-care bits in the EPCS1 device.

Read Silicon ID Operation

The read silicon ID operation code is b'1010 1011 and it lists the MSB first. Only EPCS1, EPCS4, EPCS16, and EPCS64 devices support this operation. This operation reads the 8-bit silicon ID of the EPCS device from the DATA output pin. If this operation is shifted in during an erase or write cycle, it is ignored and does not affect the cycle that is in progress.

Table 14 lists the EPCS device silicon IDs.

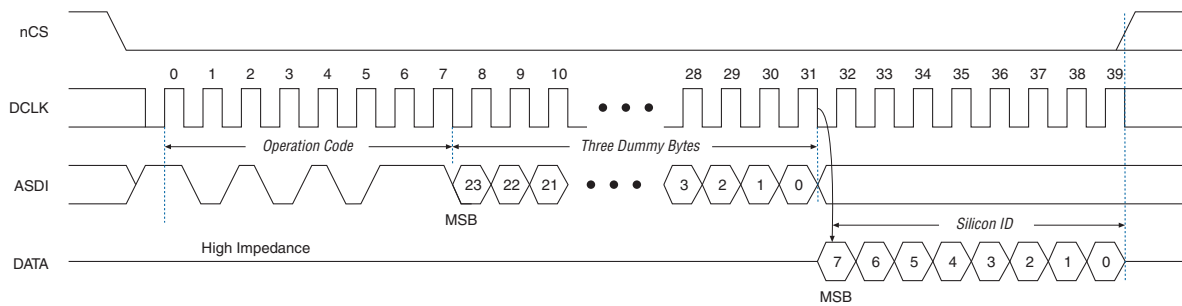
Table 14. EPCS Device Silicon ID

EPCS Device	Silicon ID (Binary Value)
EPCS1	b'0001 0000
EPCS4	b'0001 0010
EPCS16	b'0001 0100
EPCS64	b'0001 0110

The device implements the read silicon ID operation by driving the nCS signal low and then shifting in the read silicon ID operation code, followed by three dummy bytes on the ASDI pin. The 8-bit silicon ID of the EPCS device is then shifted out on the DATA pin on the falling edge of the DCLK signal. The device can terminate the read silicon ID operation by driving the nCS signal high after reading the silicon ID at least one time. Sending additional clock cycles on DCLK while nCS is driven low can cause the silicon ID to be shifted out repeatedly.

Figure 13 shows the instruction sequence of the read silicon ID operation.

Figure 13. Read Silicon ID Operation Timing Diagram ⁽¹⁾



Note to Figure 13:

(1) Only EPCS1, EPCS4, EPCS16, and EPCS64 devices support the read silicon ID operation.

Read Device Identification Operation

The read device identification operation code is b' 1001 1111 and it lists the MSB first. Only EPCS128 device supports this operation. This operation reads the 8-bit device identification of the EPCS device from the DATA output pin. If this operation is shifted in during an erase or write cycle, it is ignored and does not affect the cycle that is in progress. Table 15 lists the EPCS device identification.

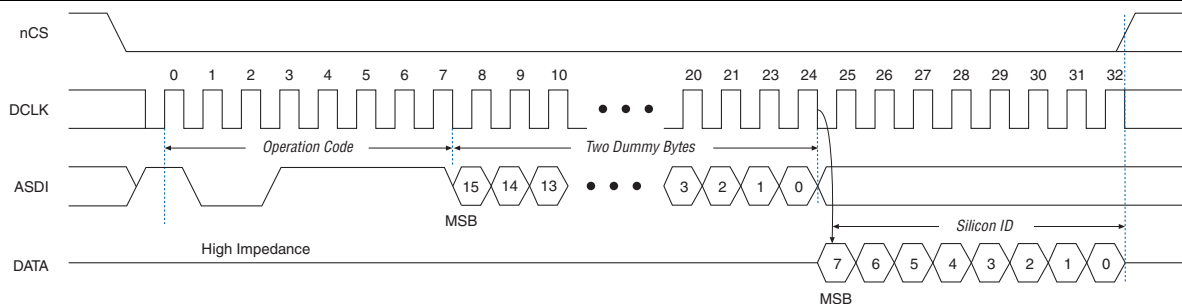
Table 15. EPCS Device Identification

EPCS Device	Silicon ID (Binary Value)
EPCS128	b' 0001 1000

The device implements the read device identification operation by driving the nCS signal low and then shifting in the read device identification operation code, followed by two dummy bytes on the ASDI pin. The 16-bit device identification of the EPCS device is then shifted out on the DATA pin on the falling edge of the DCLK signal. The device can terminate the read device identification operation by driving the nCS signal high after reading the device identification at least one time.

Figure 14 shows the instruction sequence of the read device identification operation.

Figure 14. Read Device Identification Operation Timing Diagram ⁽¹⁾



Note to Figure 14:

(1) Only EPCS128 device supports the read device identification operation.

Write Bytes Operation

The write bytes operation code is `b'0000 0010` and it lists the MSB first. This operation allows bytes to be written to the memory. You must execute the write enable operation before the write bytes operation to set the write enable latch bit in the status register to 1.

The write bytes operation is implemented by driving the `nCS` signal low, followed by the write bytes operation code, three address bytes, and at least one data byte on the `ASDI` pin. If the eight LSBs (`A[7..0]`) are not all 0, all sent data that goes beyond the end of the current page is not written into the next page. Instead, this data is written at the start address of the same page (from the address whose eight LSBs are all 0). You must ensure the `nCS` signal is set low during the entire write bytes operation.

If more than 256 data bytes are shifted into the EPCS device with a write bytes operation, the previously latched data is discarded and the last 256 bytes are written to the page. However, if less than 256 data bytes are shifted into the EPCS device, they are guaranteed to be written at the specified addresses and the other bytes of the same page are not affected.

If your design requires writing more than 256 data bytes to the memory, more than one page of memory is required. Send the write enable and write bytes operation codes, followed by three new targeted address bytes and 256 data bytes, before a new page is written.

The `nCS` signal must be driven high after the eighth bit of the last data byte has been latched in. Otherwise, the device does not execute the write bytes operation. The write enable latch bit in the status register is reset to 0 before the completion of each write bytes operation. Therefore, the write enable operation must be carried out before the next write bytes operation.

The device initiates a self-timed write cycle immediately after the `nCS` signal is driven high. For more information about the self-timed write cycle time, refer to the t_{WB} value in [Table 16 on page 29](#). You must account for this amount of delay before another page of memory is written. Alternatively, you can check the write in progress bit in the status register by executing the read status operation while the self-timed write cycle is in progress. The write in progress bit is set to 1 during the self-timed write cycle and 0 when it is complete.



You must erase all the memory bytes of the EPCS devices to all 1 or `0xFF` before you implement the write bytes operation. You can erase all the memory bytes by executing the erase sector operation in a sector or the erase bulk operation throughout the entire memory.