Order Number: AN2006
Rev. 1, 6/2001

Freescale Semiconductor, Inc.

Application Note

Interfacing the ColdFire® MCF5206e to a 1 Mbit x 16 SDRAM

Nigel Dick
East Kilbride, Scotland

Due to advances in technology, synchronous DRAMs offer higher speeds, density, and performance than conventional asynchronous DRAMs. The ability of SDRAMs to connect seamlessly to many processors combined with the rapid decrease in cost makes it an attractive choice for many embedded application designs.

This document attempts to determine the cost effectiveness of interfacing a standard “off-the-shelf” synchronous DRAM with the MCF5206e asynchronous DRAM controller via some form of low-cost programmable logic device with minimal additional logic.

The ColdFire MCF5206eLITE evaluation board comes complete with a standard asynchronous DRAM controller and an onboard 72-pin SIMM socket for supporting varying types of ADRAM SIMMs.

1.1 Hardware Considerations

In order to determine whether the MCF5206e is a less expensive solution than upgrading to the MCF5307 (which has an on-chip synchronous/asynchronous DRAM controller), note the approximate cost, as follows:

5206e unit cost = $12.47
5307 unit cost = $20.51

Given the costs, the glue logic that processes and generates the control signals for the SDRAM must be less than $8 (unit cost) to be cost-effective. Selection of the programmable logic device is limited to one of many CMOS 22 input, 10 output PALs available. The unit cost of these devices ranges from $5 to $9, depending

© Freescale Semiconductor, Inc., 2004. All rights reserved.
© Motorola, Inc., 2001. All rights reserved.

on the manufacturer and the propagation delay of the chip. Immediately, the overall cost is limited, confining the SDRAM controlling logic into a very small PAL.

Due to these limitations, a Vantis PALLV22V10 (22 input/10 output, low voltage) device and a relatively small Micron MT48LC1M16A1 (1 Mbit x 16) SDRAM was chosen. The functional block diagram shown in Figure 1 shows the architecture of the MT48LC1M16A1, hereafter referred to simply as SDRAM.

As shown in Figure 1, the SDRAM's memory is arranged in two banks controlled directly by the chip control logic and the logic level on the BA input pin. The control logic performs the synchronization between all input and output signals with the system clock and hence makes the control interface much simpler.

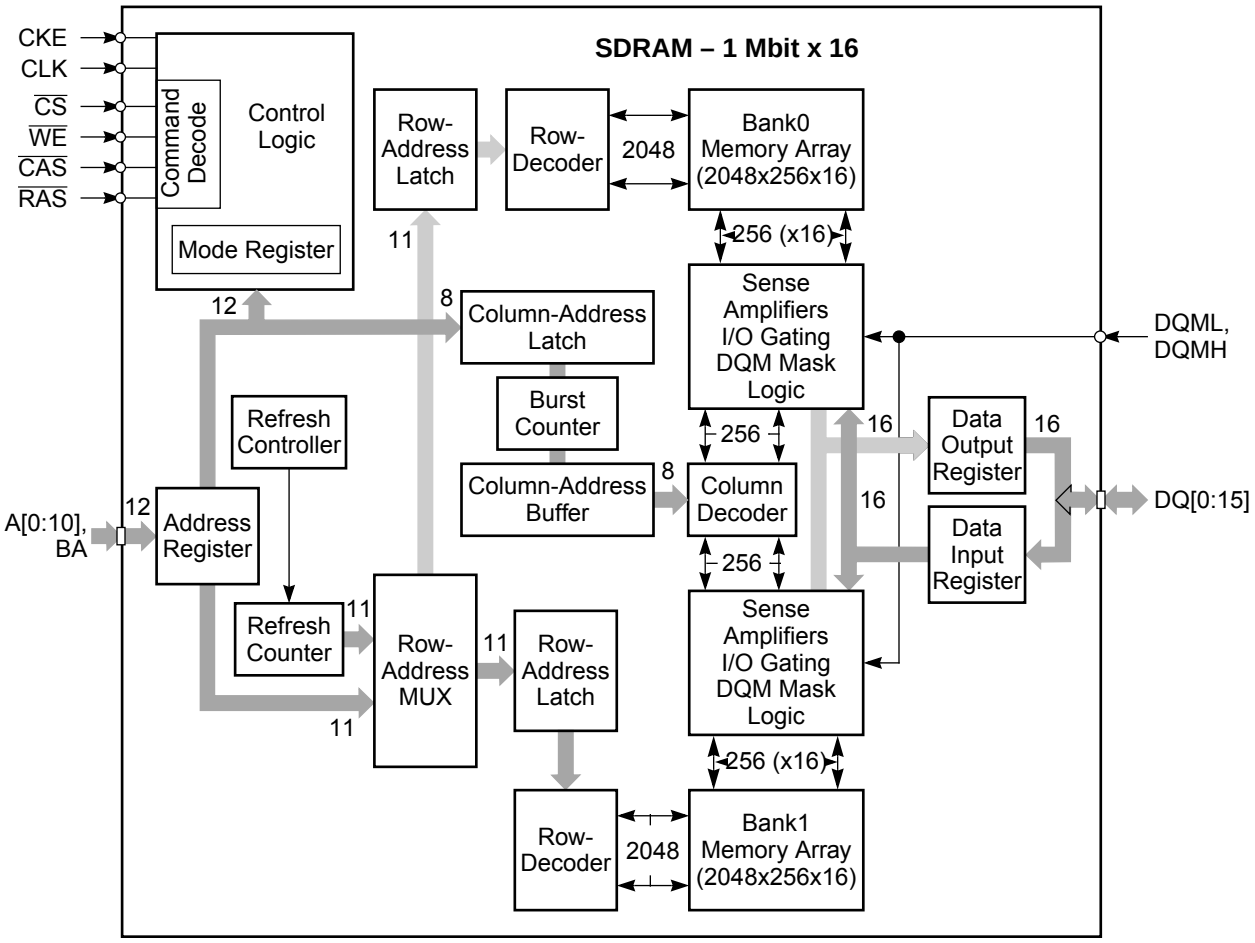

Figure 1. Functional Block Diagram—1 Mbit x 16 SDRAM

Since data can only be read from or written to one bank at any given time, one bank can be precharged while the other bank is accessed. This process is known as interleaving. In order to keep the design minimal, however, this example uses the SDRAM's auto-precharge and auto-refresh modes of operation (see Section 1.2, "SDRAM Commands").

1.2 SDRAM Commands

Each of the commands performed by the SDRAM are generated through the main control inputs to the device (!CAS, !RAS, !WE, !CS) and the more specialized chip control inputs BA, A10, and CKE. Table 1

For More Information On This Product,
Go to: www.freescale.com

Freescale Semiconductor, Inc.

shows how the commands are formed from the input pins and then gives a brief summary of the main instructions.

Table 1. SDRAM Commands

Command Name	!CS	!RAS	!CAS	!WE	DQM	DQ	Notes
INHIBIT	H	X	X	X	X	X	
NOP (IDLE)	L	H	H	X	X	X	
ACTIVE	L	L	H	X	Bank/Row	X	1
READ	L	H	L	6	Bank/Col	X	2
WRITE	L	H	L	6	Bank/Col	X	2
PRECHARGE	L	L	H	X	Code	X	3
AUTO REFRESH	L	L	L	X	Code	X	4
LOAD MODE REG	L	L	H	X	Code	X	5

Notes:

1. BA[x] determines which bank is made active; A0–A10 determine the row address during ACTIVE command phase.
2. A10 selects the auto precharge feature (only valid for this example). BA determines which bank is read from or written to. A0–A7 provide the column address during the READ/WRITE phase.
3. A10 high determines that all banks are precharged (only valid for this example) during the READ/WRITE phase.
4. CKE is tied high (in this example) to enable auto refresh.
5. A0–A10 and BA inputs de ne the opcode that is wr itten into the mode register.
6. DQs are activated or deactivated during reads (2-clock delay) and writes (zero clock delay).

1.3 Glossary of Commands

COMMAND INHIBIT — This command prevents the SDRAM from executing new commands regardless of whether or not the clock is enabled. Operations currently being executed are not affected. This command deselects the SDRAM.

NOP (IDLE) — The NOP command prevents the SDRAM from executing any instructions during wait or idle states. Again, operations being executed are not affected.

LOAD MODE REGISTER — This command can only be executed when all banks are idle and is used to determine addressing mode, burst length and CAS latency. It is only used during the initialization stage of the SDRAM.

ACTIVE — The ACTIVE command is used to open access to a particular row prior to an access. The row remains open until the row is precharged. A precharge command must also be issued prior to writing to another row in the same bank. Input BA is required to select the relevant bank and address inputs select the row.

AUTO REFRESH — This command is used during normal operation of an SDRAM (usually $\overline{CAS}$ before $\overline{RAS}$ signals); however, AUTO REFRESH must be carried out when it is required because it is non-persistent. Input CKE determines

For More Information On This Product,
Go to: www.freescale.com

Freescale Semiconductor, Inc.

	whether the SDRAM is in self-refresh or auto-refresh mode. In this example, to reduce complexity, CKE is tied high to set the SDRAM into auto-refresh mode.
PRECHARGE	The precharge command is used to deactivate an active row or to de-activate all of the rows in all of the banks. On this particular SDRAM, input A10 determines whether one bank or all of the banks are precharged. Again, to reduce complexity, this command is valid only for A10 input logic high (that is, precharge both banks). Again, this command is used during the initial configuration of the SDRAM.
READ	The READ command is used to initiate an active read burst to an active row. The logic level on the bank input pin determines the bank that is written to and the state of A10 input determines whether auto precharge is enabled or disabled. Since, for this example, A10 is always logic high during the READ command phase, it selects auto precharge and therefore each row being accessed is precharged at the end of each read burst. Read data appears on the DQ inputs subject to the level on the inputs 2 clock cycles earlier.
WRITE	The WRITE command is used to initiate an active write burst to a valid row. Again, BA and A10 inputs select the specific bank and also whether auto-refresh is enabled or disabled. If any DQ input is registered low, the corresponding data will be written to memory. If any DQ input is registered high, the corresponding data is ignored.

1.4 SDRAM Timing Considerations

Prior to use, the SDRAM must first be initialized as follows: (Note: Timing values refer to -8A version (125 MHz clock).)

1. Apply power to the device and generate a stable clock. A delay of at least 100µs is required prior to issuing any command. The only valid instructions at this time are NOP and COMMAND INHIBIT.
2. Place all banks in the idle state before use by precharging both banks for a time no less than 24 ns (tRP).
3. While the bank is in the idle state, at least 8 auto refresh cycles must execute with a minimum delay of 80 ns (tRC) between each one. Repeat the two auto refresh command wakeups whenever the 64 ms (tREF) requirement is exceeded.
4. Set the mode register to program $\overline{\text{CAS}}$ latency, burst length, and write mode. A delay of 2 clock cycles (tMRD) is required prior to the next activate or refresh.
5. Activate a row before accessing it. BA input selects the bank, A0–A10 selects the row. A delay of 24 ns (tRCD) is required before any READ or WRITE command during the ACTIVE command phase.

If these steps are not followed, the device may not operate as defined in this document.

In order to determine the input signals being sent from the MCF5206e to the PAL, each possible transitional waveform diagram from the MCF5206e must be carefully analyzed in terms of $\overline{\text{RAS}}$, $\overline{\text{CAS}}$, DRAMW, and CS. By carefully analyzing each waveform, it is then possible to create the correct assignment logic generated by the PAL which in turn drives the appropriate control/command word to the SDRAM. See Figure 2.

Freescale Semiconductor, Inc.

For More Information On This Product,
Go to: www.freescale.com

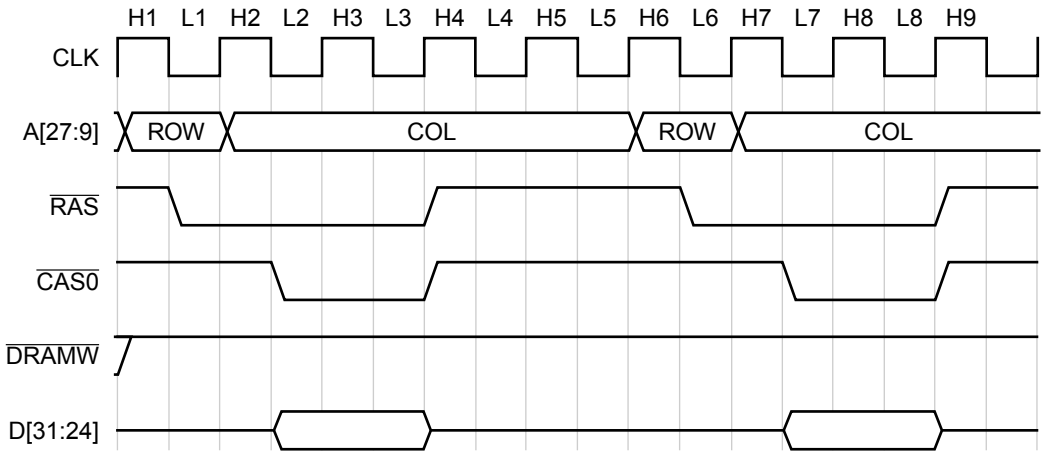

Figure 2. Byte Read Transfers in Normal Mode with 8-bit DRAM

The basic timing diagram for a standard 8-bit DRAM read (Figure 2) shows that data is valid only when $\overline{RAS}$ and $\overline{CAS}$ are asserted at the same time that DRAMW is not asserted (all signals are active low, DRAMW asserted is a write). So, from this observation, the condition for a move into state READ on the PAL state diagram is true if $\overline{CAS}$ & !DRAMW (where ! equates to not asserted). $\overline{CAS}$ becomes non-asserted at the end of a valid data byte.

Again, from these observations, the state transition diagram remains in state READ as long as $\overline{CAS}$ is not asserted. Since we need to know which bank is being read or written and also the logic state of address line A10, these inputs form the return truth logic path for returning back to the active state. Assuming we are returning back to the active state which has bank 1 active and address A10 is logic 0 (ActiveHL), the return loop requires !CAS & BA & !A10 to be true to make the transition.

In a similar fashion to creating the state transition equations for the byte read operation, the same process can be carried out for writing data to the SDRAM. Figure 3 shows the timing waveform for a long-word write to a 16-bit SDRAM.

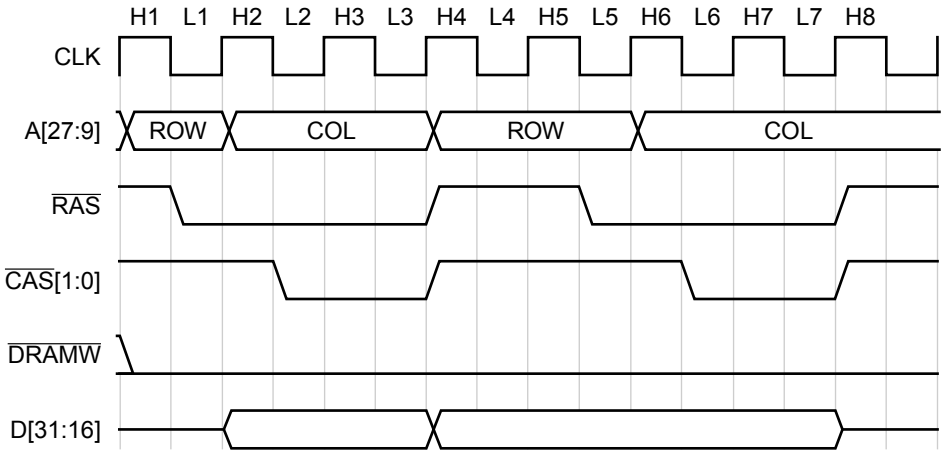

Figure 3. Longword Write Transfer in Normal Mode with 16-bit SDRAM

To generate a write to an SDRAM location the processor needs to supply $\overline{RAS}$ and $\overline{CAS}$ signals asserted as before and also assert the DRAMW input to signify a write. Data then becomes valid when $\overline{CAS}$ is asserted on the falling edge of the clock at position H2. Therefore, to transition to a WRITE state in the state transition diagram, the following logic equation must be true: DRAMW & CAS & BA & A10 (this assumes

For More Information On This Product,
Go to: www.freescale.com

Freescale Semiconductor, Inc.

moving from the ACTIVEHH state where address A10 is logic high and the data to be written is in bank 1, BA is logic high).

An important feature of the SDRAM is the need for refreshing the memory during normal operations. As discussed previously, for this example, the device was configured in the auto-refresh mode. Figure 4 shows the typical $\overline{CAS}$ before $\overline{RAS}$ refresh timing waveforms.

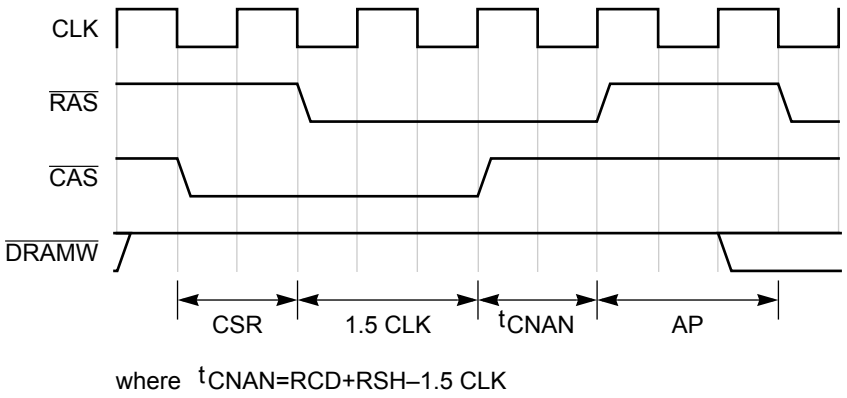

Figure 4. $\overline{CAS}$ before $\overline{RAS}$ Refresh Cycle Timing

The more simplistic timing diagram of the $\overline{CAS}$ before $\overline{RAS}$ refresh of the SDRAM involves a slightly more complex formulation of the state transition diagram which requires the use of a dummy state variable. The first transition of the diagram to the dummy state occurs if $\overline{CAS}$ is asserted. The transition to auto refresh occurs if $\overline{RAS}$ then becomes asserted.

The complete state transition diagram can be created following a similar strategy for the remaining timing scenarios; see Section 1.7, “Logic State Diagram & PAL Equations,” for more information.

1.5 Proposed Hardware Configuration

Figure 5 shows a possible configuration of the MCF5206e SDRAM interface. In the diagram, signals are tapped from the address lines to feed both the A10 and BA inputs to the PAL. A10 connects directly to the A10 input on the PAL and the BA input connects to the most significant address line (since there is only one BA input pin, logic 0 on this pin selects bank 0 and logic 1 selects bank 1).

$\overline{CAS}$ inputs (CAS0IN and CAS1IN) are fed into the PAL since only two $\overline{CAS}$ signals are needed for a 16-bit SDRAM. These signals are used to generate the column addressing for the SDRAM. The clock enable pin on the PAL is also tied to VDD to minimize decode logic by the PAL which effectively hardwires the PAL into performing auto- rather than self-refreshing mode of operation.

Freescale Semiconductor, Inc.

For More Information On This Product,
Go to: www.freescale.com

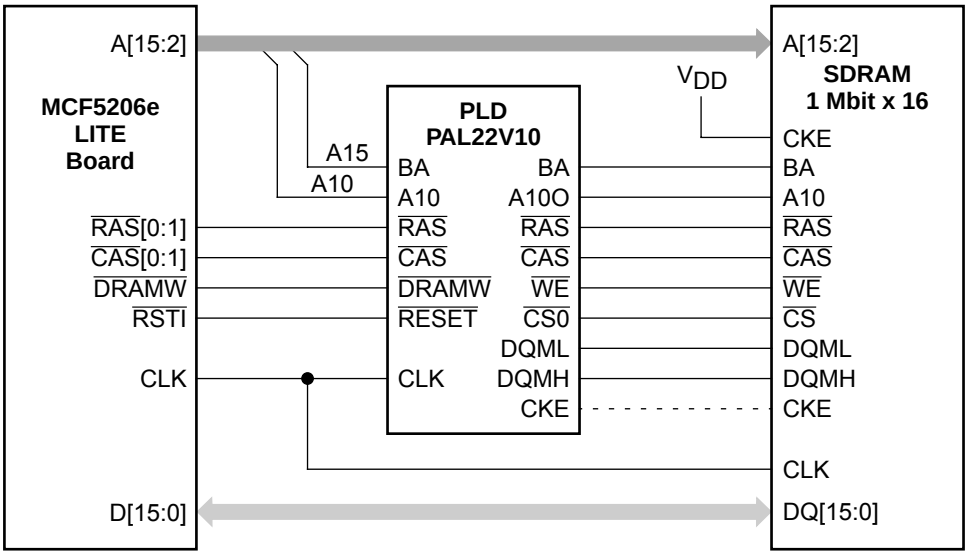

Figure 5. Proposed MCF5206e SDRAM Interface

The clock output signal from the MCF5206e supplies the system clock for the interface. This is fed directly to both the PAL and the SDRAM chips. Select the type of PAL22V10 so that propagation delay between input and clocked output is minimal. If the propagation delay is too great, a race condition will exist between the control signals (such as $\overline{CAS}$, $\overline{RAS}$, $\overline{WE}$) and the main data and address buses which feed directly to the SDRAM from the MCF5206e. So note that another trade off is speed—the faster the device, the greater the cost.

For the purpose of proving that a low-cost PAL can physically decode the logic required, it has been assumed that the propagation delay is negligible and that a 6 ns access device (the -8A speed rated part) is adequate. The proposed solution in Figure 5 is by no means the only solution and is extremely basic with no additional clock or input buffers being facilitated. Different strategies will obviously create different logic equations and hence use a different number of available gates on the PAL.

1.6 Proposed Software Configuration

In order to create easy transitions from different states, it is necessary to predefine the structure of the software. By implementing dummy reads and writes at particular instances in the executed code, it is easy to perform delay loops within a state diagram.

Recalling the necessary sequence for initialization detailed in Section 1.4, “SDRAM Timing Considerations,” with respect to delay times and command sequences, the following piece of code can be written to perform the initialization sequence (offset should be set so that A10–A0 = $220. Offset would also take into account the value programmed into MBAR registers):

```
**********************************
; Example code to initialize SDRAM chip and program load mode register via
; A10-A0. Note that the value programmed into load mode register ($220) will
; configure it for single location access, standard operation, CAS latency = 2,
; sequential burst (interleave is disabled through the sole use of auto precharge
; for this example), and a burst length of 1.
.
**********************************
```

INITIALIZATION:

For More Information On This Product,
Go to: www.freescale.com

Freescale Semiconductor, Inc.

```
NOP                    ; Kill time waiting on initialization
LEA     Offset+$220, A0; Configure A10 high, CAS latency=2 & config the
MOVE.B  #$06,D0        ; load mode register. Store dummy value
MOVE.B  D0,(A0)        ; Perform dummy write.(state:precharge both banks)
NOP                    ; Precharge delay e.g nop (min 24ns)
MOVE.B  #$06,D0        ; Command ignored, effectively a NOP.
MOVE.B   (A0),D0       ; Perform dummy read.(state:Autorefresh cycle #1)
MOVE.B  #$06,D0        ; Command ignored, effectively a NOP.
MOVE.B   (A0),D0       ; Perform dummy read.(state:Autorefresh cycle #2)
MOVE.B  #$06,D0        ; Command ignored, effectively a NOP.
MOVE.B   (A0),D0       ; Perform dummy read.(state:Autorefresh cycle #3)
MOVE.B  #$06,D0        ; Command ignored, effectively a NOP.
MOVE.B   (A0),D0       ; Perform dummy read.(state:Autorefresh cycle #4)
MOVE.B  #$06,D0        ; Command ignored, effectively a NOP.
MOVE.B   (A0),D0       ; Perform dummy read.(state:Autorefresh cycle #5)
MOVE.B  #$06,D0        ; Command ignored, effectively a NOP.
MOVE.B   (A0),D0       ; Perform dummy read.(state:Autorefresh cycle #6)
MOVE.B  #$06,D0        ; Command ignored, effectively a NOP.
MOVE.B   (A0),D0       ; Perform dummy read.(state:Autorefresh cycle #7)
MOVE.B  #$06,D0        ; Command ignored, effectively a NOP.
MOVE.B   (A0),D0       ; Perform dummy read.(state:Autorefresh cycle #8)
MOVE.B  #$06,D0        ; Command ignored, effectively a NOP.
MOVE.B   (A0),D0       ; Perform dummy read.(state:Autorefresh cycle #9)
MOVE.B  #$06,D0        ; Command ignored, effectively a NOP.
MOVE.B  D0,(A0)        ; Perform write. (state:Load mode Register) Value
                       ; written into register is taken from A10-A0.
MOVE.B  #$06,D0        ; Command ignored, effectively a NOP.
MOVE.B   (A0),D0       ; Perform dummy read.(state:idle)

; This would complete the initialization sequence prior to actual row
   activation!
```

Similarly, the software should conform to each true logic state for a state transition in the final diagram. Since the ACTIVE command requires inputs from BA (to determine the bank) and A10 (to select the appropriate column address), four different activate states are required.

1.7 Logic State Diagram & PAL Equations

Using the waveform diagrams from the MCF5206e processor and the correct SDRAM command instructions, the state diagram can be constructed as shown in Figure 6. Note that, for simplification, the read/write section of the state diagram shows only the state for ActiveHL (activate row with BA input high and A10 input low). The complete state diagram contains similar conditional loops for both ActiveHH, ActiveLL and ActiveLH, where HH, LL, and LH refer to the states on BA and A10 respectively.

Freescale Semiconductor, Inc.

For More Information On This Product,
Go to: www.freescale.com

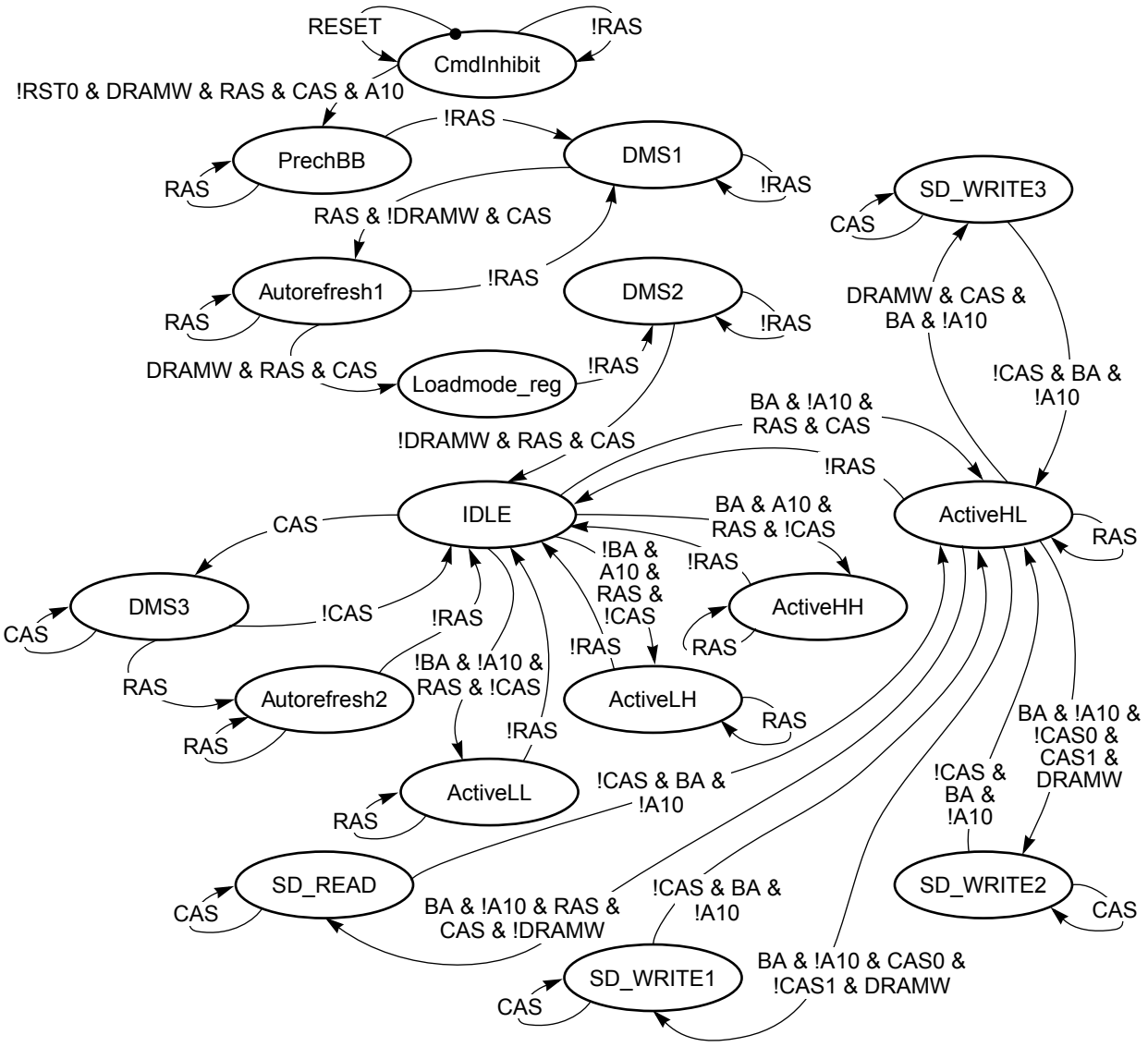

Figure 6. Simplified Version of SDRAM Controller State Diagram (ActiveHL)

It is easy to construct the final PAL logic equations and pin assignments using the state diagram shown in Figure 6 and command equations for the SDRAM. Section Appendix A, “P22V10 PAL File,” shows the generated PAL file (the conversion format of the file has been coded for CUPL).

Using CUPL (with the -C option) allows the output to be spawned into PALASM format which gives an immediate indication of the number of product states required for the final solution on the PAL. The logic for every command state is made up of a combination of product terms that are taken directly from the state diagram. The PAL is constructed using a series of AND gates fed by the inputs which are combined together via OR gates to make up the product terms at the output pins of the device. The generated PALASM formatted output file is shown in Section Appendix B, “P22V10 PALASM Output File.”

1.8 Conclusion

This document analyzed the cost effectiveness of an interface solution connecting the MCF5206e to an external SDRAM using a PAL22V10 target programmable logic device (22 inputs and 10 outputs) and the

For More Information On This Product,
Go to: www.freescale.com

Freescale Semiconductor, Inc.

state machine needed only 8 active output states. For all intents and purposes, it seemed as though this design should work. The limiting factor, however, is the number of available internal AND and OR gates within the PAL. Each output is created through a combination of product terms of the inputs and it is these product terms that are quickly exhausted.

After compiling the SDRAM PAL file using CUPL, the error message shows there is an insufficient number of product terms available within the PAL to create the logic required by the SDRAM logic file. Appendix A, “P22V10 PAL File,” shows the CUPL SDRAM PAL file. By enabling the output to spawn a PALASM file, it is apparent the number of product terms needed for each output pin, shown in Appendix B, “P22V10 PALASM Output File.”

It is immediately apparent that because of the large number of output states required by the state machine, the number of product terms is very large (>40). Obviously, the low-cost PAL simply does not have enough gates to produce such a large number of product terms.

In order to create the logic required to interface an SDRAM with the MCF5206e, the programmable logic controller would require a PAL with more internal gates than the PALLV22V10 for full implementation. It would therefore be more cost effective for the customer to purchase a MCF5307 that seamlessly interfaces to the SDRAM rather than implement some interface logic in the form of a PAL.

A bigger PAL would cost more money which would not make the solution cost effective (for example, using some form of EPLD). Altera currently produce a 32 macrocell EPLD which allows up to 68 product terms which costs 9.14 (GBP) per unit. Add this to the cost of a standard MCF5206e processor and the system cost would be greater than simply buying an MCF5307.

Freescale Semiconductor, Inc.

For More Information On This Product,
Go to: www.freescale.com

Appendix A
P22V10 PAL File

```
Name      PAL_SD;
Partno    P00001;
Date      28th Jun 99;
Revision  00;
Designer  Nigel Dick (EKB);
Company   Copyright (C);
Assembly  5206e_lite_board;
Location  Uxx;
Device    P22V10;

/*************************************************************/
/* This device generates the necessary intialisation    */
/* and timing logic to allow interfacing between the    */
/* Freescale 5206e Lite board and a Micron (TM) 16Mb: x16*/
/* 1T SDRAM using a Vantis PAL22V10 chip.               */
/*************************************************************/
/** Allowable Target Device Types : PAL22V10 sdip       **/
/*************************************************************/
/**  Inputs  **/

PIN 1        = CLK                  ;    /* System Bus clock              */
PIN 2        = !RESET               ;    /* Main system reset             */
PIN 3        = !RST0                ;    /* Interrupt 0 signal from 5206e */
PIN [4..5] = ![CAS0IN,CAS1IN]       ;    /* CAS0/CAS1 from 5206e          */
PIN [6..7] = ![RAS0IN,RAS1IN]       ;    /* RAS0/RAS1 from 5206e          */
PIN 8        = BA                   ;    /* Bank Selection:MSB of address */
PIN 9        = A10                  ;    /* Address A10 / special function */
PIN 10       = !DRAMW               ;    /* DRAM read / write             */

/**  Outputs  **/

PIN 14       = BAO                  ;     /* DRAM Bank address selection   */
PIN 15       = A10O                 ;     /* DRAM A10 address/ special fnct */
PIN 16       = DQMH                 ;     /* DRAM data input valid high    */
PIN 17       = DQML                 ;     /* DRAM data input valid low    */
PIN 18       = !WEO                 ;     /* DRAM write enable             */
PIN 19       = !CASO                ;     /* DRAM column address strobe    */
PIN 20       = !RASO                ;     /* DRAM row address strobe       */
PIN 21       = !CSO                 ;     /* DRAM chip select              */

/*************************************************************/
/** Declarations and Intermediate Variable Definitions **/
/**                                                   **/
/**  - Define all valid states of the DRAM logic state **/
/**    machine.                                       **/
/*************************************************************/
```


For More Information On This Product,
Go to: www.freescale.com

Freescale Semiconductor, Inc.

```
Field DRAM_SM = [ CSO, RASO, CASO, WEO, DQML, DQMH, A10O, BAO ] ;
$define DRAM_LOADMR     'b'00001100
$define DRAM_AUTORFSH1  'b'00011101
$define DRAM_AUTORFSH2  'b'00011100
$define DRAM_PRECHBB    'b'00101111
$define DRAM_ACTIVEHH   'b'00111111
$define DRAM_ACTIVEHL   'b'00111110
$define DRAM_ACTIVELL   'b'00111100
$define DRAM_ACTIVELH   'b'00111101
$define DRAM_READH      'b'01010001
$define DRAM_READL      'b'01010000
$define DRAM_WRITE1H    'b'01001011
$define DRAM_WRITE1L    'b'01001010
$define DRAM_WRITE2L    'b'01000110
$define DRAM_WRITE2H    'b'01000111
$define DRAM_WRITE3H    'b'01000011
$define DRAM_WRITE3L    'b'01000010
$define DRAM_IDLE       'b'01111111
$define DRAM_DMS1       'b'01111110
$define DRAM_DMS2       'b'01111101
$define DRAM_DMS3       'b'01111011
$define DRAM_CMDINH     'b'11111111


/**  Logic Equations  **/

/
*************************************************************************
****/
/**            DRAM Logic Controller, state machine summary
**/
/**
**/
/** DRAM_LOADMR= Load mode register A10, BA =logic 0                    **/
/** DRAM_AUTORFSH1= Auto refresh state, initialisation stage only         **/
/** DRAM_AUTORFSH2= Auto refresh state, CAS before RAS refresh            **/
/** DRAM_PRECHBB= Precharge both banks at initialisation, A10 = logic 1 **/
/** DRAM_ACTIVEHH= Select bank / activate row. A10=logic 1, BA=logic 1   **/
/** DRAM_ACTIVEHL= Select bank / activate row. A10=logic 1, BA=logic 0   **/
/** DRAM_ACTIVELL= Select bank / activate row. A10=logic 0, BA=logic 0   **/
/** DRAM_ACTIVELH= Select bank / activate row. A10=logic 0, BA=logic 1   **/
/** DRAM_READH= Read valid data. Auto precharge enabled (A10=logic 1)   **/
/**             BA selects valid bank, BA=logic 1.                      **/
/** DRAM_READL= Read valid data. Auto precharge enabled (A10=logic 1)   **/
/**             BA selects valid bank, BA=logic 0.                      **/
/** DRAM_WRITE1H= Write valid data. Auto precharge enabled (A10=logic1) **/
/**             BA selects valid bank, BA=logic 1.                      **/
/** DRAM_WRITE1L= Write valid data. Auto precharge enabled (A10=logic1) **/
/**             BA selects valid bank, BA=logic 0.                      **/
/** DRAM_WRITE2L= Write valid data. Auto precharge enabled (A10=logic1) **/
```


For More Information On This Product,
Go to: www.freescale.com

Freescale Semiconductor, Inc.

```
/**            BA selects valid bank, BA=logic 0.                      **/
/** DRAM_WRITE2H= Write valid data. Auto precharge enabled (A10=logic1) **/
/**            BA selects valid bank, BA=logic 1.                      **/
/** DRAM_WRITE3H= Write valid data. Auto precharge enabled (A10=logic1) **/
/**            BA selects valid bank, BA=logic 1.                      **/
/** DRAM_WRITE3L= Write valid data. Auto precharge enabled (A10=logic1) **/
/**            BA selects valid bank, BA=logic 0.                      **/
/** DRAM_IDLE= DRAM logic in idle, no operation (NOP) mode.            **/
/** DRAM_DMS1= DRAM dummy state 1. Used for delay tRP.                 **/
/** DRAM_DMS2= DRAM dummy state 2. Used for delay tMRD.                **/
/** DRAM_DMS3= DRAM dummy state 3. Used for decoding CAS/RAS refresh   **/
/** DRAM_CMDINH- DRAM in the command inhibit mode.
**/
/
*************************************************************************
****/

sequence DRAM_SM   {

present DRAM_CMDINH if RESET                              next    DRAM_CMDINH;
                    if !RASIN                             next    DRAM_CMDINH;
                    if A10 & !RST0 & RASIN & CASIN & DRAMWnext   DRAM_PRECHBB;

present DRAM_PRECHBBif RASIN                              next    DRAM_PRECHBB;
                    if !RASIN                             next    DRAM_DMS1;

present DRAM_DMS1   if !RASIN                             next    DRAM_DMS1;
                    if RASIN & !DRAMW & CASIN             next    DRAM_AUTORFSH1;

present DRAM_AUTORFSH1  if RASIN                          next    DRAM_AUTORFSH1;
                        if !RASIN                         next    DRAM_DMS1;
                        if RASIN & CASIN & DRAMW          next    DRAM_LOADMR;

present DRAM_LOADMR if !RASIN                             next    DRAM_DMS2;

present DRAM_DMS2   if !RASIN                             next    DRAM_DMS2;
                    if RASIN & CASIN & !DRAMW             next    DRAM_IDLE;

present DRAM_IDLE   if CASIN                              next    DRAM_DMS3;
                    if !A10 & !BA & RASIN & !CASIN        next    DRAM_ACTIVELL;
                    if !A10 & BA & RASIN & !CASIN         next    DRAM_ACTIVELH;
                    if A10 & !BA & RASIN & !CASIN         next    DRAM_ACTIVEHL;
                    if A10 & BA & RASIN & !CASIN          next    DRAM_ACTIVEHH;

present DRAM_DMS3   if CASIN                              next    DRAM_DMS3;
                    if !CASIN                             next    DRAM_IDLE;
                    if RASIN                              next    DRAM_AUTORFSH2;

present DRAM_AUTORFSH2if RASIN                            next    DRAM_AUTORFSH2;
                       if !RASIN                          next    DRAM_IDLE;
```


For More Information On This Product,
Go to: www.freescale.com

Freescale Semiconductor, Inc.

P22V10 PAL File

```
/** Need to determine active state, dependant on A10 and BA inputs ! **/

present DRAM_ACTIVELLif !RASIN                              next    DRAM_IDLE;
                    if RASIN                                next    DRAM_ACTIVELL;
                    if !A10 & !BA & CASIN & RASIN & !DRAMWnext    DRAM_READL;
                    if !A10 & !BA & CASIN & RASIN & !DRAMWnext    DRAM_READH;
                    if !A10 & !BA & CAS0IN & !CAS1IN & DRAMWnext
DRAM_WRITE1L;
                    if !A10 & !BA & !CAS0IN & CAS1IN & DRAMWnext
DRAM_WRITE2L;
                    if !A10 & !BA & CASIN & DRAMW       next    DRAM_WRITE3L;

present DRAM_ACTIVELHif !RASIN                              next    DRAM_IDLE;
                    if RASIN                                next    DRAM_ACTIVELH;
                    if !A10 & BA & CASIN & RASIN & !DRAMWnext    DRAM_READL;
                    if !A10 & BA & CASIN & RASIN & !DRAMWnext    DRAM_READH;
                    if !A10 & BA & CAS0IN & !CAS1IN & DRAMWnext
DRAM_WRITE1L;
                    if !A10 & BA & !CAS0IN & CAS1IN & DRAMWnext
DRAM_WRITE2L;
                    if !A10 & BA & CASIN & DRAMW        next    DRAM_WRITE3L;

present DRAM_ACTIVEHLif !RASIN                              next    DRAM_IDLE;
                    if RASIN                                next    DRAM_ACTIVEHL;
                    if A10 & !BA & CASIN & RASIN & !DRAMWnext    DRAM_READL;
                    if A10 & !BA & CASIN & RASIN & !DRAMWnext    DRAM_READH;
                    if A10 & !BA & CAS0IN & !CAS1IN & DRAMWnext
DRAM_WRITE1L;
                    if A10 & !BA & !CAS0IN & CAS1IN & DRAMWnext
DRAM_WRITE2L;
                    if A10 & !BA & CASIN & DRAMW        next    DRAM_WRITE3L;

present DRAM_ACTIVEHH   if !RASIN                           next    DRAM_IDLE;
                    if RASIN                                next    DRAM_ACTIVEHH;
                    if A10 & BA & CASIN & RASIN & !DRAMWnext    DRAM_READL;
                    if A10 & BA & CASIN & RASIN & !DRAMWnext    DRAM_READH;
                    if A10 & BA & CAS0IN & !CAS1IN & DRAMWnext   DRAM_WRITE1L;
                    if A10 & BA & !CAS0IN & CAS1IN & DRAMWnext   DRAM_WRITE2L;
                    if !A10 & !BA & CASIN & DRAMW       next    DRAM_WRITE3L;

/** Need to determine which block is to be written to, dependant on BA input!  **/

present DRAM_WRITE1Lif CAS0IN & !BA & DRAMW                 next    DRAM_WRITE1L;
                    if !CAS0IN & !A10 & !BA                 next    DRAM_ACTIVELL;
                    if !CAS0IN & !A10 & BA                  next    DRAM_ACTIVELH;
                    if !CAS0IN & A10 & !BA                  next    DRAM_ACTIVEHL;
                    if !CAS0IN & A10 & BA                   next    DRAM_ACTIVEHH;

present DRAM_WRITE2Lif CAS1IN & !BA & DRAMW                 next    DRAM_WRITE2L;
                    if !CAS1IN & !A10 & !BA                 next    DRAM_ACTIVELL;
                    if !CAS1IN & !A10 & BA                  next    DRAM_ACTIVELH;
```

Freescale Semiconductor, Inc.

For More Information On This Product,
Go to: www.freescale.com

```
                    if !CAS1IN & A10 & !BA              next    DRAM_ACTIVEHL;
                    if !CAS1IN & A10 & BA               next    DRAM_ACTIVEHH;

present DRAM_WRITE3Lif CASIN & !BA & DRAMW              next    DRAM_WRITE3L;
                    if !CASIN & !A10 & !BA              next    DRAM_ACTIVELL;
                    if !CASIN & !A10 & BA               next    DRAM_ACTIVELH;
                    if !CASIN & A10 & !BA               next    DRAM_ACTIVEHL;
                    if !CASIN & A10 & BA                next    DRAM_ACTIVEHH;

present DRAM_WRITE1Hif CAS0IN & BA & DRAMW              next    DRAM_WRITE1H;
                    if !CAS0IN & !A10 & !BA             next    DRAM_ACTIVELL;
                    if !CAS0IN & !A10 & BA              next    DRAM_ACTIVELH;
                    if !CAS0IN & A10 & !BA              next    DRAM_ACTIVEHL;
                    if !CAS0IN & A10 & BA               next    DRAM_ACTIVEHH;

present DRAM_WRITE2Hif CAS1IN & BA & DRAMW              next    DRAM_WRITE2H;
                    if !CAS1IN & !A10 & !BA             next    DRAM_ACTIVELL;
                    if !CAS1IN & !A10 & BA              next    DRAM_ACTIVELH;
                    if !CAS1IN & A10 & !BA              next    DRAM_ACTIVEHL;
                    if !CAS1IN & A10 & BA               next    DRAM_ACTIVEHH;

present DRAM_WRITE3Hif CASIN & BA & DRAMW               next    DRAM_WRITE3H;
                    if !CASIN & !A10 & !BA              next    DRAM_ACTIVELL;
                    if !CASIN & !A10 & BA               next    DRAM_ACTIVELH;
                    if !CASIN & A10 & !BA               next    DRAM_ACTIVEHL;
                    if !CASIN & A10 & BA                next    DRAM_ACTIVEHH;

/** Need to determine which block is to be read from, dependant on BA input!   **/

present DRAM_READL  if CASIN & !BA & !DRAMW             next    DRAM_READL;
                    if !CASIN & !A10 & !BA              next    DRAM_ACTIVELL;
                    if !CASIN & !A10 & BA               next    DRAM_ACTIVELH;
                    if !CASIN & A10 & !BA               next    DRAM_ACTIVEHL;
                    if !CASIN & A10 & BA                next    DRAM_ACTIVEHH;

present DRAM_READH  if CASIN & BA & !DRAMW              next    DRAM_READH;
                    if !CASIN & !A10 & !BA              next    DRAM_ACTIVELL;
                    if !CASIN & !A10 & BA               next    DRAM_ACTIVELH;
                    if !CASIN & A10 & !BA               next    DRAM_ACTIVEHL;
                    if !CASIN & A10 & BA                next    DRAM_ACTIVEHH;
}

/** Other definitions **/

/** Generate CAS for every CAS line from the 5206e **/

CASIN = CAS0IN & CAS1IN & !RESET     ;

/** Generate RAS for every RAS line from the 5206e **/
```

Freescale Semiconductor, Inc.

P22V10 PAL File

```
RASIN = RAS0IN & RAS1IN & !RESET     ;

/** Resets **/

BAO.ar  =       'b'0;
A10O.ar =       'b'0;
DQMH.ar =       'b'0;
DQML.ar =       'b'0;
WEO.ar  =       'b'0;
CASO.ar =       'b'0;
RASO.ar =       'b'0;
CSO.ar  =       'b'0;

/** Presets **/

BAO.sp  =       'b'0;
A10O.sp =       'b'0;
DQMH.sp =       'b'0;
DQML.sp =       'b'0;
WEO.sp  =       'b'0;
CASO.sp =       'b'0;
RASO.sp =       'b'0;
CSO.sp  =       'b'0;
```

Freescale Semiconductor, Inc.

Appendix B
P22V10 PALASM Output File

```
TITLE           PAL_SD
PATTERN         P00001
REVISION        00
DATE            28th Jun 99
AUTHOR          Nigel Dick (EKB)
COMPANY         Copyright (C)

CHIP pal_sd PAL22V10
CLK /RESET /RST0 /CAS0IN /CAS1IN /RAS0IN /RAS1IN BA A10 /DRAMW
NC GND NC BAO A10O DQMH DQML /WEO /CASO /RASO /CSO NC NC VCC

EQUATIONS

A10O.RSTF = GND

A10O := A10O  * BAO  * CASO  * CSO  * DQMH  * DQML  * RASO  * RESET  * WEO
     + A10O  * BAO  * CASO  * CSO  * DQMH  * DQML  * /RAS0IN  * RASO  * WEO
     + A10O  * BAO  * CASO  * CSO  * DQMH  * DQML  * /RAS1IN  * RASO  * WEO
     + A10  * A10O  * BAO  * CAS0IN  * CAS1IN  * CASO  * CSO  * DQMH  * DQML  *
DRAMW  * RAS0IN  * RAS1IN  * RASO  * /RESET  * /RST0  * WEO
     + A10O  * BAO  * CASO  * /CSO  * DQMH  * DQML  * RAS0IN  * RAS1IN  * /RASO
* /RESET
     + A10O  * BAO  * CASO  * /CSO  * DQMH  * DQML  * /RAS0IN  * /RASO
     + A10O  * BAO  * CASO  * /CSO  * DQMH  * DQML  * /RASO  * RESET
     + A10O  * BAO  * CASO  * /CSO  * DQMH  * DQML  * /RAS1IN  * /RASO
     + A10O  * /BAO  * CASO  * /CSO  * DQMH  * DQML  * /RAS0IN  * WEO
     + A10O  * /BAO  * CASO  * /CSO  * DQMH  * DQML  * RESET  * WEO
     + A10O  * /BAO  * CASO  * /CSO  * DQMH  * DQML  * /RAS1IN  * WEO
     + A10  * /A10O  * /CASO  * /CSO  * /DQMH  * /DQML  * RASO  * RESET  * WEO
     + A10  * /A10O  * /CAS1IN  * /CASO  * /CSO  * /DQMH  * /DQML  * RASO  * WEO
     + A10  * /A10O  * /CAS0IN  * /CASO  * /CSO  * /DQMH  * /DQML  * RASO  * WEO
     + A10O  * BA  * BAO  * CAS0IN  * CAS1IN  * /CASO  * /CSO  * /DQMH  * /DQML
* DRAMW  * RASO  * /RESET  * /WEO
     + /A10O  * BAO  * CAS0IN  * CAS1IN  * CASO  * /CSO  * DQMH  * DQML  * /
DRAMW  * RAS0IN  * RAS1IN  * RASO  * /RESET  * WEO
     + A10O  * BAO  * CAS0IN  * CAS1IN  * CASO  * /CSO  * DQML  * RASO  * /RESET
* WEO
     + A10  * A10O  * BAO  * /CAS0IN  * CASO  * /CSO  * DQMH  * DQML  * RAS0IN
* RAS1IN  * RASO  * /RESET  * WEO
     + A10  * A10O  * BAO  * /CAS1IN  * CASO  * /CSO  * DQMH  * DQML  * RAS0IN
* RAS1IN  * RASO  * /RESET  * WEO
     + A10O  * BAO  * /CAS0IN  * CASO  * /CSO  * /DQMH  * DQML  * RASO  * WEO
     + A10O  * BAO  * CASO  * /CSO  * /DQMH  * DQML  * RASO  * RESET  * WEO
     + A10O  * BAO  * /CAS1IN  * CASO  * /CSO  * /DQMH  * DQML  * RASO  * WEO
     + /A10O  * /CSO  * DQMH  * DQML  * /RAS0IN  * /RASO  * WEO
     + /A10O  * /CSO  * DQMH  * DQML  * /RASO  * RESET  * WEO
     + /A10O  * /CSO  * DQMH  * DQML  * /RAS1IN  * /RASO  * WEO
     + /A10  * /A10O  * /BA  * /BAO  * CAS0IN  * /CAS1IN  * CASO  * /CSO  * DQMH
```

Freescale Semiconductor, Inc.

For More Information On This Product,
Go to: www.freescale.com

```
* DQML  * DRAMW  * /RASO  * WEO
     + /A10  * /A10O  * /BA  * /BAO  * /CAS0IN  * CAS1IN  * CASO  * /CSO  * DQMH
* DQML  * DRAMW  * /RASO  * WEO
     + /A10  * /A10O  * /BA  * /BAO  * CAS0IN  * CAS1IN  * CASO  * /CSO  * DQMH
* DQML  * DRAMW  * /RASO  * /RESET  * WEO
     + /A10  * /A10O  * BA  * BAO  * CAS0IN  * /CAS1IN  * CASO  * /CSO  * DQMH
* DQML  * DRAMW  * /RASO  * WEO
     + /A10  * /A10O  * BA  * BAO  * /CAS0IN  * CAS1IN  * CASO  * /CSO  * DQMH
* DQML  * DRAMW  * /RASO  * WEO
     + /A10  * /A10O  * BA  * BAO  * CAS0IN  * CAS1IN  * CASO  * /CSO  * DQMH
* DQML  * DRAMW  * /RASO  * /RESET  * WEO
     + A10O  * /BAO  * CASO  * /CSO  * DQMH  * DQML  * RAS0IN  * RAS1IN  * /RASO
* /RESET  * WEO
     + A10  * A10O  * /BA  * /BAO  * CAS0IN  * /CAS1IN  * CASO  * /CSO  * DQMH
* DQML  * DRAMW  * /RASO  * WEO
     + A10  * A10O  * /BA  * /BAO  * /CAS0IN  * CAS1IN  * CASO  * /CSO  * DQMH
* DQML  * DRAMW  * /RASO  * WEO
     + A10  * A10O  * /BA  * /BAO  * CAS0IN  * CAS1IN  * CASO  * /CSO  * DQMH
* DQML  * DRAMW  * /RASO  * /RESET  * WEO
     + A10  * A10O  * BA  * BAO  * CAS0IN  * /CAS1IN  * CASO  * /CSO  * DQMH  *
DQML  * DRAMW  * /RASO  * WEO
     + A10  * A10O  * BA  * BAO  * /CAS0IN  * CAS1IN  * CASO  * /CSO  * DQMH  *
DQML  * DRAMW  * /RASO  * WEO
     + /A10  * A10O  * /BA  * BAO  * CAS0IN  * CAS1IN  * CASO  * /CSO  * DQMH
* DQML  * DRAMW  * /RASO  * /RESET  * WEO
     + A10O  * /BA  * /BAO  * CAS0IN  * /CASO  * /CSO  * /DQMH  * DQML  * DRAMW
* RASO  * /WEO
     + A10O  * BA  * BAO  * CAS1IN  * /CASO  * /CSO  * DQMH  * /DQML  * DRAMW
* RASO  * /WEO
     + A10O  * /BA  * /BAO  * CAS1IN  * /CASO  * /CSO  * DQMH  * /DQML  * DRAMW
* RASO  * /WEO
     + A10  * A10O  * /CAS1IN  * /CASO  * /CSO  * DQMH  * /DQML  * RASO  * /WEO
     + A10O  * /BA  * /BAO  * CAS0IN  * CAS1IN  * /CASO  * /CSO  * /DQMH  * /
DQML  * DRAMW  * RASO  * /RESET  * /WEO
     + A10  * A10O  * /CAS0IN  * /CASO  * /CSO  * /DQMH  * /DQML  * RASO  * /WEO
     + A10  * A10O  * /CAS1IN  * /CASO  * /CSO  * /DQMH  * /DQML  * RASO  * /WEO
     + A10  * A10O  * /CASO  * /CSO  * /DQMH  * /DQML  * RASO  * RESET  * /WEO
     + A10O  * BA  * BAO  * CAS0IN  * /CASO  * /CSO  * /DQMH  * DQML  * DRAMW
* RASO  * /WEO
     + A10  * A10O  * /CAS0IN  * /CASO  * /CSO  * /DQMH  * DQML  * RASO  * /WEO

A10O.CLKF = CLK

A10O.SETF = GND

BAO.RSTF = GND

BAO := A10O  * BAO  * CASO  * CSO  * DQMH  * DQML  * RASO  * RESET  * WEO
    + A10O  * BAO  * CASO  * CSO  * DQMH  * DQML  * /RAS0IN  * RASO  * WEO
    + A10O  * BAO  * CASO  * CSO  * DQMH  * DQML  * /RAS1IN  * RASO  * WEO
    + A10  * A10O  * BAO  * CAS0IN  * CAS1IN  * CASO  * CSO  * DQMH  * DQML  *
DRAMW  * RAS0IN  * RAS1IN  * RASO  * /RESET  * /RST0  * WEO
    + A10O  * BAO  * CASO  * /CSO  * DQMH  * DQML  * RAS0IN  * RAS1IN  * /RASO
* /RESET
```


For More Information On This Product,
Go to: www.freescale.com

Freescale Semiconductor, Inc.

```
    + /A10O  * BA  * BAO  * CAS0IN  * CAS1IN  * /CASO  * /CSO  * /DQMH  * /DQML
* /DRAMW  * RASO  * /RESET  * WEO
    + A10O  * /BAO  * CAS0IN  * CAS1IN  * CASO  * /CSO  * DQMH  * DQML  * /DRAMW
* RAS0IN  * RAS1IN  * RASO  * /RESET  * WEO
    + /A10O  * BAO  * /CSO  * DQMH  * DQML  * RAS0IN  * RAS1IN  * /RASO  * /
RESET  * WEO
    + /A10O  * /BAO  * /CASO  * /CSO  * DQMH  * DQML  * /RAS0IN  * /RASO
    + /A10O  * /BAO  * /CASO  * /CSO  * DQMH  * DQML  * /RASO  * RESET
    + /A10O  * /BAO  * /CASO  * /CSO  * DQMH  * DQML  * /RAS1IN  * /RASO
    + /A10O  * BAO  * CASO  * /CSO  * DQMH  * DQML  * /RAS0IN  * WEO
    + /A10O  * BAO  * CASO  * /CSO  * DQMH  * DQML  * RESET  * WEO
    + /A10O  * BAO  * CASO  * /CSO  * DQMH  * DQML  * /RAS1IN  * WEO
    + /A10O  * BAO  * CAS0IN  * CAS1IN  * CASO  * /CSO  * DQMH  * DQML  * /DRAMW
* RAS0IN  * RAS1IN  * RASO  * /RESET  * WEO
   + A10O  * BAO  * CAS0IN  * CAS1IN  * CASO  * /CSO  * DQML  * RASO  * /RESET  * WEO
    + A10O  * BA  * BAO  * /CAS0IN  * CASO  * /CSO  * DQMH  * DQML  * RAS0IN  *
RAS1IN  * RASO  * /RESET  * WEO
    + A10O  * BA  * BAO  * /CAS1IN  * CASO  * /CSO  * DQMH  * DQML  * RAS0IN  *
RAS1IN  * RASO  * /RESET  * WEO
    + A10O  * BAO  * /CAS0IN  * CASO  * /CSO  * /DQMH  * DQML  * RASO  * WEO
    + A10O  * BAO  * CASO  * /CSO  * /DQMH  * DQML  * RASO  * RESET  * WEO
    + A10O  * BAO  * /CAS1IN  * CASO  * /CSO  * /DQMH  * DQML  * RASO  * WEO
    + /BAO  * CASO  * /CSO  * DQMH  * DQML  * /RAS0IN  * /RASO  * WEO
    + /BAO  * CASO  * /CSO  * DQMH  * DQML  * /RASO  * RESET  * WEO
    + /BAO  * CASO  * /CSO  * DQMH  * DQML  * /RAS1IN  * /RASO  * WEO
    + /A10  * /A10O  * /BA  * /BAO  * CAS0IN  * CAS1IN  * CASO  * /CSO  * DQMH
* DQML  * /DRAMW  * RAS0IN  * RAS1IN  * /RASO  * /RESET  * WEO
    + A10  * A10O  * /BA  * /BAO  * CAS0IN  * CAS1IN  * CASO  * /CSO  * DQMH  *
DQML  * /DRAMW  * RAS0IN  * RAS1IN  * /RASO  * /RESET  * WEO
    + A10O  * BAO  * CASO  * /CSO  * DQMH  * DQML  * /RAS0IN  * /RASO  * WEO
    + A10O  * BAO  * CASO  * /CSO  * DQMH  * DQML  * /RASO  * RESET  * WEO
    + A10O  * BAO  * CASO  * /CSO  * DQMH  * DQML  * /RAS1IN  * /RASO  * WEO
    + /A10O  * BA  * /CASO  * /CSO  * /DQMH  * /DQML  * RASO  * RESET  * WEO
    + /A10O  * BA  * /CAS1IN  * /CASO  * /CSO  * /DQMH  * /DQML  * RASO  * WEO
    + /A10O  * BA  * /CAS0IN  * /CASO  * /CSO  * /DQMH  * /DQML  * RASO  * WEO
    + A10O  * BA  * /CAS1IN  * /CASO  * /CSO  * /DQMH  * /DQML  * RASO  * /WEO
    + A10O  * BA  * /CASO  * /CSO  * /DQMH  * /DQML  * RASO  * RESET  * /WEO
    + A10O  * BA  * BAO  * CAS0IN  * /CASO  * /CSO  * /DQMH  * DQML  * DRAMW  *
RASO  * /WEO
    + A10O  * BA  * /CAS0IN  * /CASO  * /CSO  * /DQMH  * DQML  * RASO  * /WEO
    + A10O  * BA  * BAO  * CAS1IN  * /CASO  * /CSO  * DQMH  * /DQML  * DRAMW  *
RASO  * /WEO
    + A10O  * BA  * /CAS1IN  * /CASO  * /CSO  * DQMH  * /DQML  * RASO  * /WEO
    + A10O  * BA  * BAO  * CAS0IN  * CAS1IN  * /CASO  * /CSO  * /DQMH  * /DQML
* DRAMW  * RASO  * /RESET  * /WEO
    + A10O  * BA  * /CAS0IN  * /CASO  * /CSO  * /DQMH  * /DQML  * RASO  * /WEO

BAO.CLKF = CLK

BAO.SETF = GND

CASO.RSTF = GND
```


For More Information On This Product,
Go to: www.freescale.com

Freescale Semiconductor, Inc.

```
CASO := A10O  * BAO  * CASO  * CSO  * DQMH  * DQML  * RASO  * RESET  * WEO
     + A10O  * BAO  * CASO  * CSO  * DQMH  * DQML  * /RAS0IN  * RASO  * WEO
     + A10O  * BAO  * CASO  * CSO  * DQMH  * DQML  * /RAS1IN  * RASO  * WEO
     + A10  * A10O  * BAO  * CAS0IN  * CAS1IN  * CASO  * CSO  * DQMH  * DQML  *
DRAMW  * RAS0IN  * RAS1IN  * RASO  * /RESET  * /RST0  * WEO
     + A10O  * BAO  * CASO  * /CSO  * DQMH  * DQML  * RAS0IN  * RAS1IN  * /RASO
* /RESET
     + A10O  * BAO  * CASO  * /CSO  * DQMH  * DQML  * /RAS0IN  * /RASO
     + A10O  * BAO  * CASO  * /CSO  * DQMH  * DQML  * /RASO  * RESET
     + A10O  * BAO  * CASO  * /CSO  * DQMH  * DQML  * /RAS1IN  * /RASO
     + A10O  * /BAO  * CASO  * /CSO  * DQMH  * DQML  * /RAS0IN  * WEO
     + A10O  * /BAO  * CASO  * /CSO  * DQMH  * DQML  * RESET  * WEO
     + A10O  * /BAO  * CASO  * /CSO  * DQMH  * DQML  * /RAS1IN  * WEO
     + /A10O  * /CASO  * /CSO  * /DQMH  * /DQML  * RASO  * RESET  * WEO
     + /A10O  * /CASO  * /CSO  * DQMH  * DQML  * /RAS0IN  * /RASO  * WEO
     + /A10O  * /CASO  * /CSO  * DQMH  * DQML  * /RASO  * RESET  * WEO
     + /A10O  * /CASO  * /CSO  * DQMH  * DQML  * /RAS1IN  * /RASO  * WEO
    + /A10O  * /BAO  * /CASO  * /CSO  * DQMH  * DQML  * /RAS0IN  * /RASO  * /WEO
     + /A10O  * /BAO  * /CASO  * /CSO  * DQMH  * DQML  * /RASO  * RESET  * /WEO
    + /A10O  * /BAO  * /CASO  * /CSO  * DQMH  * DQML  * /RAS1IN  * /RASO  * /WEO
     + /A10O  * BAO  * CASO  * /CSO  * DQMH  * DQML  * /RAS0IN  * WEO
     + /A10O  * BAO  * CASO  * /CSO  * DQMH  * DQML  * RESET  * WEO
     + /A10O  * BAO  * CASO  * /CSO  * DQMH  * DQML  * /RAS1IN  * WEO
      + /A10O  * BAO  * CAS0IN  * CAS1IN  * CASO  * /CSO  * DQMH  * DQML  * /
DRAMW  * RAS0IN  * RAS1IN  * RASO  * /RESET  * WEO
     + A10O  * BAO  * CAS0IN  * CAS1IN  * CASO  * /CSO  * DQML  * RASO  * /RESET
* WEO
     + /A10O  * /CAS1IN  * /CASO  * /CSO  * /DQMH  * /DQML  * RASO  * WEO
     + /A10O  * /CAS0IN  * /CASO  * /CSO  * /DQMH  * /DQML  * RASO  * WEO
    + A10O  * BAO  * /CAS0IN  * CASO  * /CSO  * DQMH  * DQML  * RAS0IN  * RAS1IN
* RASO  * /RESET  * WEO
    + A10O  * BAO  * /CAS1IN  * CASO  * /CSO  * DQMH  * DQML  * RAS0IN  * RAS1IN
* RASO  * /RESET  * WEO
     + A10O  * BAO  * /CAS0IN  * CASO  * /CSO  * /DQMH  * DQML  * RASO  * WEO
     + A10O  * BAO  * CASO  * /CSO  * /DQMH  * DQML  * RASO  * RESET  * WEO
     + A10O  * BAO  * /CAS1IN  * CASO  * /CSO  * /DQMH  * DQML  * RASO  * WEO
     + /A10O  * /BAO  * CASO  * /CSO  * DQMH  * DQML  * /RAS0IN  * /RASO  * WEO
     + /A10O  * /BAO  * CASO  * /CSO  * DQMH  * DQML  * /RASO  * RESET  * WEO
     + /A10O  * /BAO  * CASO  * /CSO  * DQMH  * DQML  * /RAS1IN  * /RASO  * WEO
     + /A10O  * CASO  * /CSO  * DQMH  * DQML  * RAS0IN  * RAS1IN  * /RASO  * /
RESET  * WEO
     + A10O  * /BAO  * CASO  * /CSO  * DQMH  * DQML  * RAS0IN  * RAS1IN  * /RASO
* /RESET  * WEO
     + A10O  * /CASO  * /CSO  * /DQMH  * /DQML  * RASO  * RESET  * /WEO
     + A10O  * /CAS0IN  * /CASO  * /CSO  * /DQMH  * DQML  * RASO  * /WEO
     + A10O  * /CAS1IN  * /CASO  * /CSO  * /DQMH  * /DQML  * RASO  * /WEO
     + A10O  * /CAS1IN  * /CASO  * /CSO  * DQMH  * /DQML  * RASO  * /WEO
     + A10O  * /CAS0IN  * /CASO  * /CSO  * /DQMH  * /DQML  * RASO  * /WEO

CASO.CLKF = CLK
```


For More Information On This Product,
Go to: www.freescale.com

Freescale Semiconductor, Inc.

```
CASO.SETF = GND

CSO.RSTF = GND

CSO := A10O  * BAO  * CASO  * CSO  * DQMH  * DQML  * RASO  * RESET  * WEO
    + A10O  * BAO  * CASO  * CSO  * DQMH  * DQML  * /RAS0IN  * RASO  * WEO
    + A10O  * BAO  * CASO  * CSO  * DQMH  * DQML  * /RAS1IN  * RASO  * WEO

CSO.CLKF = CLK

CSO.SETF = GND

DQMH.RSTF = GND

DQMH := A10O  * BAO  * CASO  * CSO  * DQMH  * DQML  * RASO  * RESET  * WEO
     + A10O  * BAO  * CASO  * CSO  * DQMH  * DQML  * /RAS0IN  * RASO  * WEO
     + A10O  * BAO  * CASO  * CSO  * DQMH  * DQML  * /RAS1IN  * RASO  * WEO
     + A10  * A10O  * BAO  * CAS0IN  * CAS1IN  * CASO  * CSO  * DQMH  * DQML  *
DRAMW  * RAS0IN  * RAS1IN  * RASO  * /RESET  * /RST0  * WEO
     + A10O  * BAO  * CASO  * /CSO  * DQMH  * DQML  * RAS0IN  * RAS1IN  * /RASO
* /RESET
     + A10O  * BAO  * CASO  * /CSO  * DQMH  * DQML  * /RAS0IN  * /RASO
     + A10O  * BAO  * CASO  * /CSO  * DQMH  * DQML  * /RASO  * RESET
     + A10O  * BAO  * CASO  * /CSO  * DQMH  * DQML  * /RAS1IN  * /RASO
     + A10O  * /BAO  * CASO  * /CSO  * DQMH  * DQML  * /RAS0IN  * WEO
     + A10O  * /BAO  * CASO  * /CSO  * DQMH  * DQML  * RESET  * WEO
     + A10O  * /BAO  * CASO  * /CSO  * DQMH  * DQML  * /RAS1IN  * WEO
     + A10O  * /BAO  * CAS0IN  * CAS1IN  * CASO  * /CSO  * DQMH  * DQML  * /
DRAMW  * RAS0IN  * RAS1IN  * RASO  * /RESET  * WEO
     + /A10O  * /CASO  * /CSO  * /DQMH  * /DQML  * RASO  * RESET  * WEO
     + /A10O  * /CASO  * /CSO  * DQMH  * DQML  * /RAS0IN  * /RASO  * WEO
     + /A10O  * /CASO  * /CSO  * DQMH  * DQML  * /RASO  * RESET  * WEO
     + /A10O  * /CASO  * /CSO  * DQMH  * DQML  * /RAS1IN  * /RASO  * WEO
    + /A10O  * /BAO  * /CASO  * /CSO  * DQMH  * DQML  * /RAS0IN  * /RASO  * /WEO
     + /A10O  * /BAO  * /CASO  * /CSO  * DQMH  * DQML  * /RASO  * RESET  * /WEO
    + /A10O  * /BAO  * /CASO  * /CSO  * DQMH  * DQML  * /RAS1IN  * /RASO  * /WEO
     + /A10O  * BAO  * CASO  * /CSO  * DQMH  * DQML  * /RAS0IN  * WEO
     + /A10O  * BAO  * CASO  * /CSO  * DQMH  * DQML  * RESET  * WEO
     + /A10O  * BAO  * CASO  * /CSO  * DQMH  * DQML  * /RAS1IN  * WEO
     + /A10O  * BAO  * CAS0IN  * CAS1IN  * CASO  * /CSO  * DQMH  * DQML  * /
DRAMW  * RAS0IN  * RAS1IN  * RASO  * /RESET  * WEO
     + /A10O  * /CAS1IN  * /CASO  * /CSO  * /DQMH  * /DQML  * RASO  * WEO
     + /A10O  * /CAS0IN  * /CASO  * /CSO  * /DQMH  * /DQML  * RASO  * WEO
     + A10O  * BA  * BAO  * CAS1IN  * /CASO  * /CSO  * DQMH  * /DQML  * DRAMW
* RASO  * /WEO
     + A10O  * BAO  * /CAS0IN  * CASO  * /CSO  * DQMH  * DQML  * RAS0IN  * RAS1IN
* RASO  * /RESET  * WEO
     + A10O  * BAO  * /CAS1IN  * CASO  * /CSO  * DQMH  * DQML  * RAS0IN  * RAS1IN
* RASO  * /RESET  * WEO
     + A10O  * BAO  * /CAS0IN  * CASO  * /CSO  * /DQMH  * DQML  * RASO  * WEO
```

Freescale Semiconductor, Inc.

For More Information On This Product,
Go to: www.freescale.com

```
     + A10O  * BAO  * CASO  * /CSO  * /DQMH  * DQML  * RASO  * RESET  * WEO
     + A10O  * BAO  * /CAS1IN  * CASO  * /CSO  * /DQMH  * DQML  * RASO  * WEO
     + A10O  * BAO  * CASO  * /CSO  * /DQMH  * DQML  * RAS0IN  * RAS1IN  * RASO
* /RESET  * WEO
     + /A10O  * /BAO  * CASO  * /CSO  * DQMH  * DQML  * /RAS0IN  * /RASO  * WEO
     + /A10O  * /BAO  * CASO  * /CSO  * DQMH  * DQML  * /RASO  * RESET  * WEO
     + /A10O  * /BAO  * CASO  * /CSO  * DQMH  * DQML  * /RAS1IN  * /RASO  * WEO
    + /A10O  * /CSO  * DQMH  * DQML  * RAS0IN  * RAS1IN  * /RASO  * /RESET  * WEO
     + /A10  * /A10O  * /BA  * /BAO  * /CAS0IN  * CAS1IN  * CASO  * /CSO  * DQMH
* DQML  * DRAMW  * /RASO  * WEO
     + /A10  * /A10O  * BA  * BAO  * /CAS0IN  * CAS1IN  * CASO  * /CSO  * DQMH
* DQML  * DRAMW  * /RASO  * WEO
     + A10O  * /BAO  * CASO  * /CSO  * DQMH  * DQML  * RAS0IN  * RAS1IN  * /RASO
* /RESET  * WEO
     + A10  * A10O  * /BA  * /BAO  * /CAS0IN  * CAS1IN  * CASO  * /CSO  * DQMH
* DQML  * DRAMW  * /RASO  * WEO
     + A10  * A10O  * BA  * BAO  * /CAS0IN  * CAS1IN  * CASO  * /CSO  * DQMH  *
DQML  * DRAMW  * /RASO  * WEO
     + A10O  * /CASO  * /CSO  * /DQMH  * /DQML  * RASO  * RESET  * /WEO
     + A10O  * /CAS0IN  * /CASO  * /CSO  * /DQMH  * DQML  * RASO  * /WEO
     + A10O  * /BA  * /BAO  * CAS1IN  * /CASO  * /CSO  * DQMH  * /DQML  * DRAMW
* RASO  * /WEO
     + A10O  * /CAS1IN  * /CASO  * /CSO  * /DQMH  * /DQML  * RASO  * /WEO
     + A10O  * /CAS1IN  * /CASO  * /CSO  * DQMH  * /DQML  * RASO  * /WEO
     + A10O  * /CAS0IN  * /CASO  * /CSO  * /DQMH  * /DQML  * RASO  * /WEO

DQMH.CLKF = CLK
DQMH.SETF = GND

DQML.RSTF = GND

DQML := A10O  * BAO  * CASO  * CSO  * DQMH  * DQML  * RASO  * RESET  * WEO
     + A10O  * BAO  * CASO  * CSO  * DQMH  * DQML  * /RAS0IN  * RASO  * WEO
     + A10O  * BAO  * CASO  * CSO  * DQMH  * DQML  * /RAS1IN  * RASO  * WEO
     + A10  * A10O  * BAO  * CAS0IN  * CAS1IN  * CASO  * CSO  * DQMH  * DQML  *
DRAMW  * RAS0IN  * RAS1IN  * RASO  * /RESET  * /RST0  * WEO
     + A10O  * BAO  * CASO  * /CSO  * DQMH  * DQML  * RAS0IN  * RAS1IN  * /RASO
* /RESET
     + A10O  * BAO  * CASO  * /CSO  * DQMH  * DQML  * /RAS0IN  * /RASO
     + A10O  * BAO  * CASO  * /CSO  * DQMH  * DQML  * /RASO  * RESET
     + A10O  * BAO  * CASO  * /CSO  * DQMH  * DQML  * /RAS1IN  * /RASO
     + A10O  * /BAO  * CASO  * /CSO  * DQMH  * DQML  * /RAS0IN  * WEO
     + A10O  * /BAO  * CASO  * /CSO  * DQMH  * DQML  * RESET  * WEO
     + A10O  * /BAO  * CASO  * /CSO  * DQMH  * DQML  * /RAS1IN  * WEO
     + A10O  * /BAO  * CAS0IN  * CAS1IN  * CASO  * /CSO  * DQMH  * DQML  * /
DRAMW  * RAS0IN  * RAS1IN  * RASO  * /RESET  * WEO
     + /A10O  * /CASO  * /CSO  * /DQMH  * /DQML  * RASO  * RESET  * WEO
     + /A10O  * /CASO  * /CSO  * DQMH  * DQML  * /RAS0IN  * /RASO  * WEO
     + /A10O  * /CASO  * /CSO  * DQMH  * DQML  * /RASO  * RESET  * WEO
     + /A10O  * /CASO  * /CSO  * DQMH  * DQML  * /RAS1IN  * /RASO  * WEO
    + /A10O  * /BAO  * /CASO  * /CSO  * DQMH  * DQML  * /RAS0IN  * /RASO  * /WEO
     + /A10O  * /BAO  * /CASO  * /CSO  * DQMH  * DQML  * /RASO  * RESET  * /WEO
```


For More Information On This Product,
Go to: www.freescale.com

Freescale Semiconductor, Inc.

```
     + /A10O  * /BAO  * /CASO  * /CSO  * DQMH  * DQML  * /RAS1IN  * /RASO  * /WEO
      + /A10O  * BAO  * CASO  * /CSO  * DQMH  * DQML  * /RAS0IN  * WEO
      + /A10O  * BAO  * CASO  * /CSO  * DQMH  * DQML  * RESET  * WEO
      + /A10O  * BAO  * CASO  * /CSO  * DQMH  * DQML  * /RAS1IN  * WEO
      + /A10O  * BAO  * CAS0IN  * CAS1IN  * CASO  * /CSO  * DQMH  * DQML  * /
DRAMW  * RAS0IN  * RAS1IN  * RASO  * /RESET  * WEO
     + A10O  * BAO  * CAS0IN  * CAS1IN  * CASO  * /CSO  * DQML  * RASO  * /RESET
* WEO
     + /A10O  * /CAS1IN  * /CASO  * /CSO  * /DQMH  * /DQML  * RASO  * WEO
     + /A10O  * /CAS0IN  * /CASO  * /CSO  * /DQMH  * /DQML  * RASO  * WEO
     + A10O  * BAO  * /CAS0IN  * CASO  * /CSO  * DQMH  * DQML  * RAS0IN  * RAS1IN
* RASO  * /RESET  * WEO
     + A10O  * BAO  * /CAS1IN  * CASO  * /CSO  * DQMH  * DQML  * RAS0IN  * RAS1IN
* RASO  * /RESET  * WEO
     + A10O  * BAO  * /CAS0IN  * CASO  * /CSO  * /DQMH  * DQML  * RASO  * WEO
     + A10O  * BAO  * CASO  * /CSO  * /DQMH  * DQML  * RASO  * RESET  * WEO
     + A10O  * BAO  * /CAS1IN  * CASO  * /CSO  * /DQMH  * DQML  * RASO  * WEO
     + A10O  * BAO  * CASO  * /CSO  * /DQMH  * DQML  * RAS0IN  * RAS1IN  * RASO
* /RESET  * WEO
     + /A10O  * /BAO  * CASO  * /CSO  * DQMH  * DQML  * /RAS0IN  * /RASO  * WEO
     + /A10O  * /BAO  * CASO  * /CSO  * DQMH  * DQML  * /RASO  * RESET  * WEO
     + /A10O  * /BAO  * CASO  * /CSO  * DQMH  * DQML  * /RAS1IN  * /RASO  * WEO
     + /A10O  * /CSO  * DQMH  * DQML  * RAS0IN  * RAS1IN  * /RASO  * /RESET  * WEO
      + A10O  * BA  * BAO  * CAS0IN  * /CASO  * /CSO  * /DQMH  * DQML  * DRAMW
* RASO  * /WEO
     + /A10  * /A10O  * /BA  * /BAO  * CAS0IN  * /CAS1IN  * CASO  * /CSO  * DQMH
* DQML  * DRAMW  * /RASO  * WEO
     + /A10  * /A10O  * BA  * BAO  * CAS0IN  * /CAS1IN  * CASO  * /CSO  * DQMH
* DQML  * DRAMW  * /RASO  * WEO
     + A10O  * /BAO  * CASO  * /CSO  * DQMH  * DQML  * RAS0IN  * RAS1IN  * /RASO
* /RESET  * WEO
     + A10  * A10O  * /BA  * /BAO  * CAS0IN  * /CAS1IN  * CASO  * /CSO  * DQMH
* DQML  * DRAMW  * /RASO  * WEO
     + A10  * A10O  * BA  * BAO  * CAS0IN  * /CAS1IN  * CASO  * /CSO  * DQMH  *
DQML  * DRAMW  * /RASO  * WEO
     + A10O  * /BA  * /BAO  * CAS0IN  * /CASO  * /CSO  * /DQMH  * DQML  * DRAMW
* RASO  * /WEO
     + A10O  * /CASO  * /CSO  * /DQMH  * /DQML  * RASO  * RESET  * /WEO
     + A10O  * /CAS0IN  * /CASO  * /CSO  * /DQMH  * DQML  * RASO  * /WEO
     + A10O  * /CAS1IN  * /CASO  * /CSO  * /DQMH  * /DQML  * RASO  * /WEO
     + A10O  * /CAS1IN  * /CASO  * /CSO  * DQMH  * /DQML  * RASO  * /WEO
     + A10O  * /CAS0IN  * /CASO  * /CSO  * /DQMH  * /DQML  * RASO  * /WEO

DQML.CLKF = CLK

DQML.SETF = GND

RASO.RSTF = GND

RASO := A10O  * BAO  * CASO  * CSO  * DQMH  * DQML  * RASO  * RESET  * WEO
     + A10O  * BAO  * CASO  * CSO  * DQMH  * DQML  * /RAS0IN  * RASO  * WEO
     + A10O  * BAO  * CASO  * CSO  * DQMH  * DQML  * /RAS1IN  * RASO  * WEO
```

Freescale Semiconductor, Inc.

For More Information On This Product,
Go to: www.freescale.com

```
     + /A10O  * BA  * BAO  * CAS0IN  * CAS1IN  * /CASO  * /CSO  * /DQMH  * /DQML
* /DRAMW  * RASO  * /RESET  * WEO
     + A10O  * BAO  * CASO  * /CSO  * DQMH  * DQML  * /RAS0IN  * /RASO
     + A10O  * BAO  * CASO  * /CSO  * DQMH  * DQML  * /RASO  * RESET
     + A10O  * BAO  * CASO  * /CSO  * DQMH  * DQML  * /RAS1IN  * /RASO
     + A10O  * /BAO  * CASO  * /CSO  * DQMH  * DQML  * /RAS0IN  * WEO
     + A10O  * /BAO  * CASO  * /CSO  * DQMH  * DQML  * RESET  * WEO
     + A10O  * /BAO  * CASO  * /CSO  * DQMH  * DQML  * /RAS1IN  * WEO
     + /A10O  * /CASO  * /CSO  * DQMH  * DQML  * /RAS0IN  * /RASO  * WEO
     + /A10O  * /CASO  * /CSO  * DQMH  * DQML  * /RASO  * RESET  * WEO
     + /A10O  * /CASO  * /CSO  * DQMH  * DQML  * /RAS1IN  * /RASO  * WEO
    + /A10O  * /BAO  * /CASO  * /CSO  * DQMH  * DQML  * /RAS0IN  * /RASO  * /WEO
     + /A10O  * /BAO  * /CASO  * /CSO  * DQMH  * DQML  * /RASO  * RESET  * /WEO
    + /A10O  * /BAO  * /CASO  * /CSO  * DQMH  * DQML  * /RAS1IN  * /RASO  * /WEO
     + /A10O  * BAO  * CASO  * /CSO  * DQMH  * DQML  * /RAS0IN  * WEO
     + /A10O  * BAO  * CASO  * /CSO  * DQMH  * DQML  * RESET  * WEO
     + /A10O  * BAO  * CASO  * /CSO  * DQMH  * DQML  * /RAS1IN  * WEO
      + /A10O  * BAO  * CAS0IN  * CAS1IN  * CASO  * /CSO  * DQMH  * DQML  * /
DRAMW  * RAS0IN  * RAS1IN  * RASO  * /RESET  * WEO
     + A10O  * BAO  * CAS0IN  * CAS1IN  * CASO  * /CSO  * DQML  * RASO  * /RESET
* WEO
     + A10O  * BAO  * /CAS0IN  * CASO  * /CSO  * /DQMH  * DQML  * RASO  * WEO
     + A10O  * BAO  * CASO  * /CSO  * /DQMH  * DQML  * RASO  * RESET  * WEO
     + A10O  * BAO  * /CAS1IN  * CASO  * /CSO  * /DQMH  * DQML  * RASO  * WEO
    + /A10O  * /BAO  * CASO  * /CSO  * DQMH  * DQML  * /RAS0IN  * /RASO  * WEO
     + /A10O  * /BAO  * CASO  * /CSO  * DQMH  * DQML  * /RASO  * RESET  * WEO
     + /A10O  * /BAO  * CASO  * /CSO  * DQMH  * DQML  * /RAS1IN  * /RASO  * WEO
     + /A10  * /A10O  * /BA  * /BAO  * CAS0IN  * CAS1IN  * CASO  * /CSO  * DQMH
* DQML  * /DRAMW  * RAS0IN  * RAS1IN  * /RASO  * /RESET  * WEO
     + /A10  * /A10O  * /BA  * /BAO  * CAS0IN  * /CAS1IN  * CASO  * /CSO  * DQMH
* DQML  * DRAMW  * /RASO  * WEO
     + /A10  * /A10O  * /BA  * /BAO  * /CAS0IN  * CAS1IN  * CASO  * /CSO  * DQMH
* DQML  * DRAMW  * /RASO  * WEO
     + /A10  * /A10O  * /BA  * /BAO  * CAS0IN  * CAS1IN  * CASO  * /CSO  * DQMH
* DQML  * DRAMW  * /RASO  * /RESET  * WEO
      + /A10  * /A10O  * BA  * BAO  * CAS0IN  * CAS1IN  * CASO  * /CSO  * DQMH
* DQML  * /DRAMW  * RAS0IN  * RAS1IN  * /RASO  * /RESET  * WEO
     + /A10  * /A10O  * BA  * BAO  * CAS0IN  * /CAS1IN  * CASO  * /CSO  * DQMH
* DQML  * DRAMW  * /RASO  * WEO
     + /A10  * /A10O  * BA  * BAO  * /CAS0IN  * CAS1IN  * CASO  * /CSO  * DQMH
* DQML  * DRAMW  * /RASO  * WEO
      + /A10  * /A10O  * BA  * BAO  * CAS0IN  * CAS1IN  * CASO  * /CSO  * DQMH
* DQML  * DRAMW  * /RASO  * /RESET  * WEO
     + A10  * A10O  * /BA  * /BAO  * CAS0IN  * CAS1IN  * CASO  * /CSO  * DQMH
* DQML  * /DRAMW  * RAS0IN  * RAS1IN  * /RASO  * /RESET  * WEO
     + A10  * A10O  * /BA  * /BAO  * CAS0IN  * /CAS1IN  * CASO  * /CSO  * DQMH
* DQML  * DRAMW  * /RASO  * WEO
     + A10  * A10O  * /BA  * /BAO  * /CAS0IN  * CAS1IN  * CASO  * /CSO  * DQMH
* DQML  * DRAMW  * /RASO  * WEO
      + A10  * A10O  * /BA  * /BAO  * CAS0IN  * CAS1IN  * CASO  * /CSO  * DQMH
* DQML  * DRAMW  * /RASO  * /RESET  * WEO
     + A10  * A10O  * BA  * BAO  * CAS0IN  * CAS1IN  * CASO  * /CSO  * DQMH  *
DQML  * /DRAMW  * RAS0IN  * RAS1IN  * /RASO  * /RESET  * WEO
```


For More Information On This Product,
Go to: www.freescale.com

Freescale Semiconductor, Inc.

```
     + A10  * A10O  * BA  * BAO  * CAS0IN  * /CAS1IN  * CASO  * /CSO  * DQMH  *
DQML  * DRAMW  * /RASO  * WEO
     + A10  * A10O  * BA  * BAO  * /CAS0IN  * CAS1IN  * CASO  * /CSO  * DQMH  *
DQML  * DRAMW  * /RASO  * WEO
     + /A10  * A10O  * /BA  * BAO  * CAS0IN  * CAS1IN  * CASO  * /CSO  * DQMH
* DQML  * DRAMW  * /RASO  * /RESET  * WEO
     + A10O  * /BA  * /BAO  * CAS0IN  * /CASO  * /CSO  * /DQMH  * DQML  * DRAMW
* RASO  * /WEO
     + A10O  * /BA  * /BAO  * CAS1IN  * /CASO  * /CSO  * DQMH  * /DQML  * DRAMW
* RASO  * /WEO
     + A10O  * /BA  * /BAO  * CAS0IN  * CAS1IN  * /CASO  * /CSO  * /DQMH  * /
DQML  * DRAMW  * RASO  * /RESET  * /WEO
     + A10O  * BA  * BAO  * CAS0IN  * /CASO  * /CSO  * /DQMH  * DQML  * DRAMW
* RASO  * /WEO
     + A10O  * BA  * BAO  * CAS1IN  * /CASO  * /CSO  * DQMH  * /DQML  * DRAMW
* RASO  * /WEO
     + A10O  * BA  * BAO  * CAS0IN  * CAS1IN  * /CASO  * /CSO  * /DQMH  * /DQML
* DRAMW  * RASO  * /RESET  * /WEO
     + /A10O  * /BA  * /BAO  * CAS0IN  * CAS1IN  * /CASO  * /CSO  * /DQMH  * /
DQML  * /DRAMW  * RASO  * /RESET  * WEO

RASO.CLKF = CLK

RASO.SETF = GND

WEO.RSTF = GND

WEO := A10O  * BAO  * CASO  * CSO  * DQMH  * DQML  * RASO  * RESET  * WEO
    + A10O  * BAO  * CASO  * CSO  * DQMH  * DQML  * /RAS0IN  * RASO  * WEO
    + A10O  * BAO  * CASO  * CSO  * DQMH  * DQML  * /RAS1IN  * RASO  * WEO
    + /A10O  * BA  * BAO  * CAS0IN  * CAS1IN  * /CASO  * /CSO  * /DQMH  * /DQML
* /DRAMW  * RASO  * /RESET  * WEO
    + A10O  * BAO  * CASO  * /CSO  * DQMH  * DQML  * /RAS0IN  * /RASO
    + A10O  * BAO  * CASO  * /CSO  * DQMH  * DQML  * /RASO  * RESET
    + A10O  * BAO  * CASO  * /CSO  * DQMH  * DQML  * /RAS1IN  * /RASO
    + A10O  * /BAO  * CASO  * /CSO  * DQMH  * DQML  * /RAS0IN  * WEO
    + A10O  * /BAO  * CASO  * /CSO  * DQMH  * DQML  * RESET  * WEO
    + A10O  * /BAO  * CASO  * /CSO  * DQMH  * DQML  * /RAS1IN  * WEO
    + A10O  * /BAO  * CAS0IN  * CAS1IN  * CASO  * /CSO  * DQMH  * DQML  * /DRAMW
* RAS0IN  * RAS1IN  * RASO  * /RESET  * WEO
    + /A10O  * /CASO  * /CSO  * /DQMH  * /DQML  * RASO  * RESET  * WEO
    + /A10O  * /CASO  * /CSO  * DQMH  * DQML  * /RAS0IN  * /RASO  * WEO
    + /A10O  * /CASO  * /CSO  * DQMH  * DQML  * /RASO  * RESET  * WEO
    + /A10O  * /CASO  * /CSO  * DQMH  * DQML  * /RAS1IN  * /RASO  * WEO
    + /A10O  * /BAO  * /CASO  * /CSO  * DQMH  * DQML  * /RAS0IN  * /RASO  * /WEO
    + /A10O  * /BAO  * /CASO  * /CSO  * DQMH  * DQML  * /RASO  * RESET  * /WEO
    + /A10O  * /BAO  * /CASO  * /CSO  * DQMH  * DQML  * /RAS1IN  * /RASO  * /WEO
    + /A10O  * BAO  * CASO  * /CSO  * DQMH  * DQML  * /RAS0IN  * WEO
    + /A10O  * BAO  * CASO  * /CSO  * DQMH  * DQML  * RESET  * WEO
    + /A10O  * BAO  * CASO  * /CSO  * DQMH  * DQML  * /RAS1IN  * WEO
    + /A10O  * BAO  * CAS0IN  * CAS1IN  * CASO  * /CSO  * DQMH  * DQML  * /DRAMW
* RAS0IN  * RAS1IN  * RASO  * /RESET  * WEO
   + A10O  * BAO  * CAS0IN  * CAS1IN  * CASO  * /CSO  * DQML  * RASO  * /RESET  * WEO
```


For More Information On This Product,
Go to: www.freescale.com

Freescale Semiconductor, Inc.

```
    + /A10O  * /CAS1IN  * /CASO  * /CSO  * /DQMH  * /DQML  * RASO  * WEO
    + /A10O  * /CAS0IN  * /CASO  * /CSO  * /DQMH  * /DQML  * RASO  * WEO
    + A10O  * BAO  * /CAS0IN  * CASO  * /CSO  * DQMH  * DQML  * RAS0IN  * RAS1IN
* RASO  * /RESET  * WEO
    + A10O  * BAO  * /CAS1IN  * CASO  * /CSO  * DQMH  * DQML  * RAS0IN  * RAS1IN
* RASO  * /RESET  * WEO
    + A10O  * BAO  * /CAS0IN  * CASO  * /CSO  * /DQMH  * DQML  * RASO  * WEO
    + A10O  * BAO  * CASO  * /CSO  * /DQMH  * DQML  * RASO  * RESET  * WEO
    + A10O  * BAO  * /CAS1IN  * CASO  * /CSO  * /DQMH  * DQML  * RASO  * WEO
    + A10O  * BAO  * CASO  * /CSO  * /DQMH  * DQML  * RAS0IN  * RAS1IN  * RASO
* /RESET  * WEO
    + /A10O  * /BAO  * CASO  * /CSO  * DQMH  * DQML  * /RAS0IN  * /RASO  * WEO
    + /A10O  * /BAO  * CASO  * /CSO  * DQMH  * DQML  * /RASO  * RESET  * WEO
    + /A10O  * /BAO  * CASO  * /CSO  * DQMH  * DQML  * /RAS1IN  * /RASO  * WEO
    + /A10O  * /CSO  * DQMH  * DQML  * RAS0IN  * RAS1IN  * /RASO  * /RESET  * WEO
    + A10O  * CASO  * /CSO  * DQMH  * DQML  * RAS0IN  * RAS1IN  * /RASO  * /
RESET  * WEO
    + /A10O  * /BA  * /BAO  * CAS0IN  * CAS1IN  * /CASO  * /CSO  * /DQMH  * /
DQML  * /DRAMW  * RASO  * /RESET  * WEO
    + A10O  * /CAS0IN  * /CASO  * /CSO  * /DQMH  * DQML  * RASO  * /WEO
    + A10O  * /CASO  * /CSO  * /DQMH  * /DQML  * RASO  * RESET  * /WEO
    + A10O  * /CAS1IN  * /CASO  * /CSO  * DQMH  * /DQML  * RASO  * /WEO
    + A10O  * /CAS0IN  * /CASO  * /CSO  * /DQMH  * /DQML  * RASO  * /WEO
    + A10O  * /CAS1IN  * /CASO  * /CSO  * /DQMH  * /DQML  * RASO  * /WEO

WEO.CLKF = CLK

WEO.SETF = GND

A10O.TRST = VCC

BAO.TRST = VCC

CASO.TRST = VCC

CSO.TRST = VCC

DQMH.TRST = VCC

DQML.TRST = VCC

RASO.TRST = VCC

WEO.TRST = VCC
```

Freescale Semiconductor, Inc.

Table 2. MCF5206e Processor Revision HIstory

Revision	Date	Change History
Rev 0	04/2000	Initial release
Rev 1	06/2001	Edits - Change all 16 Mbyte references to 16 Mbit. Change URL to re ect ne w title.

Freescale Semiconductor, Inc.

For More Information On This Product,
Go to: www.freescale.com

Freescale Semiconductor, Inc.

How to Reach Us:

Home Page:
www.freescale.com

E-mail:
support@freescale.com

USA/Europe or Locations Not Listed:
Freescale Semiconductor
Technical Information Center, CH370
1300 N. Alma School Road
Chandler, Arizona 85224
+1-800-521-6274 or +1-480-768-2130
support@freescale.com

Europe, Middle East, and Africa:
Freescale Halbleiter Deutschland GmbH
Technical Information Center
Schatzbogen 7
81829 Muenchen, Germany
+44 1296 380 456 (English)
+46 8 52200080 (English)
+49 89 92103 559 (German)
+33 1 69 35 48 48 (French)
support@freescale.com

Japan:
Freescale Semiconductor Japan Ltd.
Headquarters
ARCO Tower 15F
1-8-1, Shimo-Meguro, Meguro-ku,
Tokyo 153-0064
Japan
0120 191014 or +81 3 5437 9125
support.japan@freescale.com

Asia/Pacific:
Freescale Semiconductor Hong Kong Ltd.
Technical Information Center
2 Dai King Street
Tai Po Industrial Estate
Tai Po, N.T., Hong Kong
+800 2666 8080
support.asia@freescale.com

For Literature Requests Only:
Freescale Semiconductor Literature Distribution Center
P.O. Box 5405
Denver, Colorado 80217
1-800-441-2447 or 303-675-2140
Fax: 303-675-2150
LDCForFreescaleSemiconductor@hibbertgroup.com

Information in this document is provided solely to enable system and software implementers to use Freescale Semiconductor products. There are no express or implied copyright licenses granted hereunder to design or fabricate any integrated circuits or integrated circuits based on the information in this document.
Freescale Semiconductor reserves the right to make changes without further notice to any products herein. Freescale Semiconductor makes no warranty, representation or guarantee regarding the suitability of its products for any particular purpose, nor does Freescale Semiconductor assume any liability arising out of the application or use of any product or circuit, and specifically disclaims any and all liability, including without limitation consequential or incidental damages. “Typical” parameters which may be provided in Freescale Semiconductor data sheets and/or specifications can and do vary in different applications and actual performance may vary over time. All operating parameters, including “Typicals” must be validated for each customer application by customer’s technical experts. Freescale Semiconductor does not convey any license under its patent rights nor the rights of others. Freescale Semiconductor products are not designed, intended, or authorized for use as components in systems intended for surgical implant into the body, or other applications intended to support or sustain life, or for any other application in which the failure of the Freescale Semiconductor product could create a situation where personal injury or death may occur. Should Buyer purchase or use Freescale Semiconductor products for any such unintended or unauthorized application, Buyer shall indemnify and hold Freescale Semiconductor and its officers, employees, subsidiaries, affiliates, and distributors harmless against all claims, costs, damages, and expenses, and reasonable attorney fees arising out of, directly or indirectly, any claim of personal injury or death associated with such unintended or unauthorized use, even if such claim alleges that Freescale Semiconductor was negligent regarding the design or manufacture of the part.

freescale™
semiconductor