Using The MU9C1965A LANCAM® MP For Data Wider Than 128 Bits

INTRODUCTION

This Application Note describes how the MUSIC MU9C1965A LANCAM MP can be used to hold very wide data words. The MU9C1965A LANCAM MP is a 1024 x 128-bit CMOS content-addressable memory (CAM) with a 32-bit I/O. It has enhanced features, which include a dual Configuration Register set and a faster operating mode with no wait states after a no-match. Although a data length of 512 bits will be used as an example, the LANCAM MP can be used to hold any data word width. This document outlines the initialization and configuration of the CAM and the routines used to add, remove, and search for data.

EXPLANATION OF GENERAL METHOD

For some applications it is necessary to store and compare words wider than 128 bits. A MU9C1965A LANCAM MP can be used to store and search the extra-wide data words. Techniques are available that split the data word into individual 128-bit CAM words. Each CAM word is individually written to the Comparand register until the whole word has been searched for or added to the CAM. The method used for manipulating the data is very similar to the software technique for manipulating linked lists. Link lists use pointers to link items in a list. For example, in C language, data structures are linked together by declaring one of the members as a pointer. This member is made to point to the next data structure in the list. To terminate the list, the pointer of the last data structure is a NULL pointer.

In this case, we want to divide the whole data word into words that can be stored in separate CAM locations and have them linked together in some way. Using 16 of the available 128 bits as a TAG does this. This TAG is similar to the pointer used in a linked list. If the total data length is split up and stored in different CAM locations, the TAG of each CAM word is the address of the preceding segments. Table 1 shows how a 512-bit data word would be stored in the CAM memory array. Please note that the order in which the data word is split is arbitrary. The individual bits of the 512-bit word can be arranged to suit the user's own application.

Although 16 bits have been used for the TAG, this can be tailored to any particular application. Because the TAG represents an address of a CAM memory location, 16 bits would provide TAGs for over 60,000 locations. If an application does not require a table of this size then the number of bits used can be reduced to an appropriate number.

The data word is split and stored in five locations in the CAM memory array. The lower 16 bits of Segment 0 stores the address of the proceeding 112 bits of the data word. The upper 16 bits of Segment 0 are used for data. The order in which the word is split is not important and can be tailored to suit the user's specific application.

Segment 3	Segment 2	Segment 1	Segment 0	Validity	Address
<bits 111–80>	<bits 79–48>	<bits 47–16>	<bits 15–0>FFFEH	V	0263H
<bits 223–192>	<bits 191–160>	<bits 159–128>	<bits 127–112>0263H	V	0264H
<bits 335–304>	<bits 303–272>	<bits 271–240>	<bits 239–224>0264H	V	0265H
<bits 447–416>	<bits 415–384>	<bits 383–352>	<bits 351–336>0265H	V	0266H
<bits 511–480>	<bits 479–448>	00000000H	00000266H	V	0267H
xxxxxxxxH	xxxxxxxxH	xxxxxxxxH	xxxxxxxxH	E	0268H
xxxxxxxxH	xxxxxxxxH	xxxxxxxxH	xxxxxxxxH	E	0269H
xxxxxxxxH	xxxxxxxxH	xxxxxxxxH	xxxxxxxxH	E	026AH
xxxxxxxxH	xxxxxxxxH	xxxxxxxxH	xxxxxxxxH	E	026BH

Notes:

1. xxxxxxxxH = "Don't Care"
2. V = Valid CAM memory array location
3. E = Empty CAM memory array location

Table 1: A 512-Bit Data Word Stored in Five Consecutive CAM Locations

Another similarity that this scheme shares with a linked list is the use of a NULL pointer. This is used to show the end of the list. This is accomplished by using the start TAG = FFFEH. The first CAM word of each 512-bit word is given a NULL value. This NULL value is arbitrary to some extent, although it must pass the following criteria:

(a) It has to be a value greater than the number of CAM locations being used.
(b) It cannot be FFFFH as this could cause confusion when the CAM data bus is in the tri-state condition.

Thus, the value FFFEH is ideal, as it passes both of the criteria.

Padding bits are required when the total data word does not fit exactly into whole CAM words. In the case of a 512-bit data word, there will be 48 bits of the final CAM word that will be unused. If these bits were left unused, any search would have to mask the relevant bits. This adds a further complication to the process. Instead, the unused bit locations are given a pad value. This could be any arbitrary value, although it has to be constant throughout routines. In this case, 0 is chosen to fill all unused bit locations.

Initialization and Configuration of the CAM

The Initialization and Configuration routine has been constructed to configure four MU9C1965A LANCAM MPs for use in the following extra-wide comparand routines. The following explains the routine that is shown in Table 2. Lines 1 through 11 initialize the CAMs. Line 1 clears up any power-up anomalies that may be left in the part. Line 2 sets all devices to listen to the following commands. Line 3 resets the devices.

Lines 4 through 11 give a unique Page address to each CAM in the chain. Line 4 targets and sets the Page Address register of the highest-priority (lowest address value) device in the chain. Line 5 sets the Full flag on this device, forcing the next device in the chain to respond to the next set of initialization commands. This cycle of targeting the Page Address register, setting the Page Address value, and setting the Full flag is repeated until all devices in the chain have a unique Page Address value. Line 11 resets all devices, returning the Full flags to their normal function.

Lines 12 through 20 configure the Background Register set for use in the delete routine (the delete routine needs to use Mask Register 1 for compares). Line 13 configures the CAMs as 128 bits CAM, 0 bits RAM, and use Mask Register 1 for compares. The CAMs are also configured for Enhanced mode. Lines 14 to 18 initialize Mask Register 1 for use in comparing on only the TAG bits. Line 19 sets the Segment Control register to write to Segment 0 (lower 16 bits are the TAG bits), and read from Segments 0 through 3. Line 20 sets the Persistent Destination for Data Writes to the Comparand register.

Lines 21 through 24 configure the Foreground Register set for normal operation (normal compares without Mask Register 1). Line 22 configures the CAMs as 128 bits CAM, 0 bits RAM, and invoke no mask registers for compares. The CAMs are also configured for Enhanced mode. Line 23 sets the Segment Control register to write to Segments 0 through 3, and read from Segments 0 through 3. Line 24 sets the Persistent Destination for Data Writes to the Comparand register.

The CAMs have been configured to have the lower 16 bits of Segment 0 (32-bit segment) contain the TAG bits and the upper bits contain a 16-bit slice of the data word. The remaining three segments contain 32-bit slices of the overall data word. Mask Register 1 in the Background Register set has been configured to mask bits 127 – 16. This is to allow only the TAG bits located in Segment 0 to be used in any automatic compare. The bit assignments for Segment 0 are shown in Figures 1 and 2.

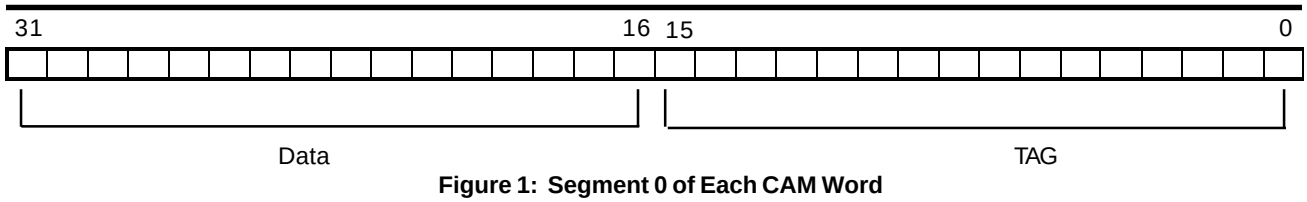


Figure 1: Segment 0 of Each CAM Word

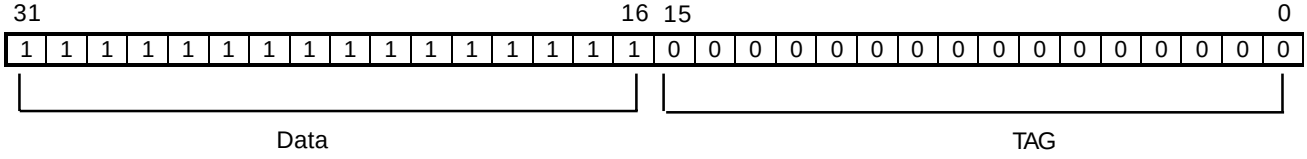


Figure 2: MR1 Segment 0 Bit Assignments

Line	/CM	/W	/EC	Cycle Length	Mnemonic	DQ[31:16]	DQ[15:0]	Description
1	L	H	H	Medium		0000H	0000H	Command Read: Clears power-up anomalies
2	L	L	H	Short	TCO_DS	0A28H	FFFFH	Select all devices
3	L	L	H	Medium	TCO_CT	0A00H	0000H	Reset all devices
4	L	L	H	Short	TCO_PA	0A08H	0000H	Set Page address = 0 for first device
5	L	L	H	Long	SFF	0700H	xxxxH	Set Full flag
6	L	L	H	Short	TCO_PA	0A08H	0001H	Set Page address =1 for second device
7	L	L	H	Long	SFF	0700H	xxxxH	Set Full flag
8	L	L	H	Short	TCO_PA	0A08H	0002H	Set Page address =2 for third device
9	L	L	H	Long	SFF	0700H	xxxxH	Set Full flag
10	L	L	H	Short	TCO_PA	0A08H	0003H	Set Page address = 3 for fourth device
11	L	L	H	Medium	TCO_CT	0A00H	0000H	Resets Full flag in all devices
12	L	L	H	Short	SBR	0619H	xxxxH	Select Background Register set
13	L	L	H	Short *	TCO_CT	0A00H	8011H	Set Control register: 128 CAM, 0 RAM, MR1, Enhanced Response mode
14	L	L	H	Short	SPD_MR1	0108H	xxxxH	Set Persistent destination to MR1
15	H	L	H	Short		FFFFH	0000H	Segment 0: TAG bits only
16	H	L	H	Short		FFFFH	FFFFH	Segment 1
17	H	L	H	Short		FFFFH	FFFFH	Segment 2
18	H	L	H	Short *		FFFFH	FFFFH	Segment 3
19	L	L	H	Short	TCO_SC	0A01H	00C0H	Set SC : Write Segment 0, Read Segments 0:3
20	L	L	H	Short	SPD_CR	0100H	xxxxH	Set Persistent destination to Comparand
21	L	L	H	Short	SFR	0618H	xxxxH	Select Foreground Register set
22	L	L	H	Short *	TCO_CT	0A00H	8001H	Set Control register: 128 CAM, 0 RAM, No Mask, Enhanced Response mode
23	L	L	H	Short	TCO_SC	0A01H	18C0H	Set SC: Write Segments 0:3, Read Segment 0:3
24	L	L	H	Short	SPD_CR	0100H	xxxxH	Set Persistent destination to Comparand

Notes:

1. xxxxH = "Don't Care"
2. Short * = This indicates cycles that are normally long, but may run as short because the results from the automatically triggered compare are not used.

Table 2: CAM Initialization and Configuration Routine

Segment 3		Segment 2		Segment 1		Segment 0	
DQ[31:16]	DQ[15:0]	DQ[31:16]	DQ[15:0]	DQ[31:16]	DQ[15:0]	DQ[31:16]	DQ[15:0]
data	data	data	data	data	data	data	TAG

Figure 3: Individual CAM Word Bit Arrangement

Adding Data Words to the CAM

Data words wider than 128 bits can be split and added to the CAM in any order. The order of the bits can be arranged to suit the user's specific application. For this example, a 512-bit data word is split into five 4-segment CAM words. These CAM words will then be written to the CAM Comparand register. Each CAM word written to the Comparand register must be of the form shown in Figure 3.

The routine in Table 3 shows how a 512-bit data word can be added to a 128-bit wide CAM. Any other data length could be implemented by adding or removing 128-bit CAM words. The last CAM word must be edited to suit by adding or removing pad bits. The software implementation of this routine, as shown in Figure 4 on page 6, simplifies using traditional looping methods.

Lines 1 through 4 write the first of the five CAM words to the Comparand register. This word has been given the start TAG FFFEH. An automatic compare is done after the last segment is written. A match can be determined from the /MF pin of the last device in the chain, or by looking at DQ31 in the Status register during the Status register read on line 5. DQ31 will be driven LOW by the highest-priority matching device, or it will be pulled HIGH by a pull-up resistor that must be added to DQ31 when using this method.

At this point a decision has to be made, based on whether a match exists or not. If a match exists, lines 6a through 6c are omitted, as there is no need to add the CAM word to the memory array. This saves memory space and simplifies any search or delete routines. If there was no-match given after the automatic compare, lines 6a through 6c are completed. Line 6b reads the Next Free Address register. This is done to find out where the CAM word will be written. Line 6c writes the CAM word to the memory array at the next free address.

Lines 7 through 10 write the next CAM word to the Comparand register. The TAG written with Segment 0 is the address where the previous CAM word can be found. This will be one of the following:

(a) If there was a match found at line 5 when the Status register was read, the TAG will be the address of the highest-priority match. This can be found by masking the Status register contents with 0000FFFFH.

(b) If there was no-match found at line 5, the TAG will be the memory array location where the CAM word is written after the move instruction. Line 6b reads from the Next Free Address register. The TAG is found by masking this with 0000FFFFH.

Lines 11 through 17 are the same as lines 5 through 11. The same procedure is followed until all CAM words have been written to the Comparand register.

Line	/CM	/W	/EC	Cycle Length	Mnemonic	DQ[31:16]	DQ[15:0]	Description
1	H	L	H	Short		<bits15–0>	FFFEH	Write seg 0 to CR (with start TAG = FFFEH)
2	H	L	H	Short		<bits 47–32>	<bits 31–16>	Write seg 1 to CR
3	H	L	H	Short		<bits 79–64>	<bits 63–48>	Write seg 2 to CR
4	H	L	L	Long		<bits 111–96>	<bits 95–80>	Write seg 3 to CR
5	L	H	H	Medium		*SR[31:16]	*SR[15:0]	Read Status reg
If a match is not found, add to memory array (otherwise omit 6a through 6c)								
6a	L	L	L	Short	TCO_NF	0218H	xxxxH	Target Next Free Address register
6b	L	H	H	Medium		xxxxH	NF[15:0]	Read NF Address reg
6c	L	L	H	Long	MOV_NF_CR_V	0334H	xxxxH	Move data from CR to NF address (set Valid)

Table 3: CAM Cycle Sequence for Adding an Entry

Line	/CM	/W	/EC	Cycle Length	Mnemonic	DQ[31:16]	DQ[15:0]	Description
7	H	L	H	Short		<bits 127–112>	ppppH	Write seg 0 to CR (with TAG = ppppH : see note)
8	H	L	H	Short		<bits 159–144>	<bits 143–128>	Write seg 1 to CR
9	H	L	H	Short		<bits 191–176>	<bits 175–160>	Write seg 2 to CR
10	H	L	L	Long		<bits 223–208>	<bits 207–192>	Write seg 3 to CR
11	L	H	H	Medium		*SR[31:16]	*SR[15:0]	Read Status reg
If a match is not found, add to memory array (otherwise omit 12a through 12c)								
12a	L	L	L	Short	TCO_NF	0218H	xxxxH	Target Next Free Address register
12b	L	H	H	Medium		xxxxH	NF[15:0]	Read NF Address reg
12c	L	L	H	Long	MOV_NF_CR_V	0334H	xxxxH	Move data from CR to NF address (set Valid)
13	H	L	H	Short		<bits 239–224>	qqqqH	Write seg 0 to CR (with TAG = qqqqH : see note)
14	H	L	H	Short		<bits 271–256>	<bits 255–240>	Write seg 1 to CR
15	H	L	H	Short		<bits 303–288>	<bits 287–272>	Write seg 2 to CR
16	H	L	L	Long		<bits 335–320>	<bits 319–304>	Write seg 3 to CR
17	L	H	H	Medium		*SR[31:16]	*SR[15:0]	Read Status reg
If a match is not found, add to memory array (otherwise omit 18a through 18c)								
18a	L	L	L	Short	TCO_NF	0218H	xxxxH	Target Next Free Address register
18b	L	H	H	Medium		xxxxH	NF[15:0]	Read NF Address reg
18c	L	L	H	Long	MOV_NF_CR_V	0334H	xxxxH	Move data from CR to NF address (set Valid)
19	H	L	H	Short		<bits 351–336>	rrrrH	Write seg 0 to CR (with TAG = rrrrH: see note)
20	H	L	H	Short		<bits 383–368>	<bits 367–352>	Write seg 1 to CR
21	H	L	H	Short		<bits 415–400>	<bits 399–384>	Write seg 2 to CR
22	H	L	L	Long		<bits 447–432>	<bits 431–416>	Write seg 3 to CR
23	L	H	H	Medium		*SR[31:16]	*SR[15:0]	Read Status reg
If a match is not found, add to memory array (otherwise omit 24a through 24c)								
24a	L	L	L	Short	TCO_NF	0218H	xxxxH	Target Next Free Address register
24b	L	H	H	Medium		xxxxH	NF[15:0]	Read NF Address reg
24c	L	L	H	Long	MOV_NF_CR_V	0334H	xxxxH	Move data from CR to NF address (set Valid)
25	H	L	H	Short		0000H	ssssH	Write seg 0 to CR (with TAG = ssssH: see note)
26	H	L	H	Short		0000H	0000H	Write seg 1 to CR
27	H	L	H	Short		<bits 479–464>	<bits 463–448>	Write seg 2 to CR
28	H	L	L	Long		<bits 511–496>	<bits 495–480>	Write seg 3 to CR

Notes:

1. xxxxH = "Don't Care"
2. *SR[31:16] , *SR[15:0] = If there was a match during the comparison, the contents of the Status register are valid and appear on DQ[31:0]. If no-match occurs, all the devices are in the tri-state condition to prevent bus contention. Pull-up resistors are required to drive the bus to 0xFFFFFFFF on a no-match condition.
3. NF[15:0] = The contents of the 16-bit Next Free Address register are valid and appear on DQ[15:0].
4. ppppH = The TAG bits for this CAM word. It is the address where the first CAM word was located or placed.
5. qqqqH = The TAG bits for this CAM word. It is the address where the second CAM word was located or placed.
6. rrrrH = The TAG bits for this CAM word. It is the address where the third CAM word was located or placed.
7. ssssH = The TAG bits for this CAM word. It is the address where the fourth CAM word was located or placed.

Table 3: CAM Cycle Sequence for Adding an Entry (*continued*)

```
void add_entry(unsigned long data_array[4][5])          /*data_array holds the 512-bit data word.
                                                          It is five CAM words and segment 0 of each
                                                        word has  0000H where the TAG will be placed */
{
unsigned long tag, status;
unsigned int location;

tag = 0x0000FFFE;                                       /* the first TAG is always FFFE */
for ( location = 0; location < 5; location++)
   {
   data_array[0][location] |= tag;                      /* add the TAG to segment 0 of each CAM word */
    dws(data_array[0][location]);                       /* data write short - first segment */
    dws(data_array[1][location]);                       /* data write short - second segment */
    dws(data_array[2][location]);                       /* data write short - third segment */
    dwlec(data_array[3][location]);                     /* data write long with /ec low - fourth segment */
    status = crm();                                     /* command read medium to read status register */
    if ( (status & 0x80000000) == 0 )/* match found in memory array (requires pull-up resistor on DQ31) */
      {
       tag = status & 0x0000FFFF;                       /* the tag for the next CAM word is the
                                                           address of the highest priority match */
      }                         /* there is no need to add the CAM word to the memory array from the CR */
     else
      {
       cwsec(0x02180000);                   /* target the NF address register for a command read.
                                               /ec must be low to unlock the daisy chain */
       tag = crm() & 0x0000FFFF;            /* command read medium - the tag for the
                                               next CAM word is the next free address */
       cwl(0x03340000);                 /* command write long - move data from CR to NF address */
      }
    }/* end for */
}/* end add_entry */
```

Figure 4: C Code for Adding an Entry

Data Search of CAM

Table 4 is a search routine to determine whether a 512-bit data word is stored in a 128-bit wide CAM. Any other data length could be implemented by adding or removing 128-bit CAM words. The last CAM word has to be edited to suit by adding or removing pad bits. The software implementation of this routine, as shown in Figure 5 on page 8, simplifies using traditional looping methods.

The search procedure involves comparing each individual CAM word with the CAM memory array. Lines 1 through 4 write the first of the five CAM words to the Comparand register. This word has been given the start TAG FFFEH. An automatic compare is done after the last segment is written. A match can be determined from the /MF pin of the last device in the chain, or by looking at DQ31 in the Status register during the Status register read on line 5. DQ31 will be driven LOW by the highest-priority matching device, or it will be pulled HIGH by a pull-up resistor that must be added to DQ31 when using this method. At this point a decision has to be made, based on whether a match exists or not:

(a) If there is no-match, the search has failed to locate a section of the 512-bit word. This means that the 512-bit word is not in the CAM. There is no need to compare the remaining CAM words.

(b) If there was a match after the automatic compare, the TAG for the next CAM word is found by masking the contents of the Status register with 0000FFFFH. This TAG is used when writing the next CAM word to the Comparand register.

This is repeated until all the CAM words have been searched for in the CAM. When the automatic compare has given a match for all five CAM words, the 512-bit word was found in the CAM.

Line	/CM	/W	/EC	Cycle Length	Mnemonic	DQ[31:16]	DQ[15:0]	Description
1	H	L	H	Short		<bits15–0>	FFFEH	Write seg 0 to CR (with start TAG = FFFEH)
2	H	L	H	Short		<bits 47–32>	<bits 31–16>	Write seg 1 to CR
3	H	L	H	Short		<bits 79–64>	<bits 63–48>	Write seg 2 to CR
4	H	L	L	Long		<bits 111–96>	<bits 95–80>	Write seg 3 to CR
5	L	H	H	Medium		*SR[31:16]	*SR[15:0]	Read Status reg
If a match was not found then the 512-bit data word was not found, therefore the search may end. If a match was found then continue.								
6	H	L	H	Short		<bits 127–112>	ppppH	Write seg 0 to CR (with TAG = ppppH: see note)
7	H	L	H	Short		<bits 159–144>	<bits 143–128>	Write seg 1 to CR
8	H	L	H	Short		<bits 191–176>	<bits 175–160>	Write seg 2 to CR
9	H	L	L	Long		<bits 223–208>	<bits 207–192>	Write seg 3 to CR
10	L	H	H	Medium		*SR[31:16]	*SR[15:0]	Read Status reg
If a match was not found then the 512-bit data word was not found, therefore the search may end. If a match was found then continue.								
11	H	L	H	Short		<bits 239–224>	qqqqH	Write seg 0 to CR (with TAG = qqqqH: see note)
12	H	L	H	Short		<bits 271–256>	<bits 255–240>	Write seg 1 to CR
13	H	L	H	Short		<bits 303–288>	<bits 287–272>	Write seg 2 to CR
14	H	L	L	Long		<bits 335–320>	<bits 319–304>	Write seg 3 to CR
15	L	H	H	Medium		*SR[31:16]	*SR[15:0]	Read Status reg
If a match was not found then the 512-bit data word was not found, therefore the search may end. If a match was found then continue.								
16	H	L	H	Short		<bits 351–336>	rrrrH	Write seg 0 to CR (with TAG = rrrrH: see note)
17	H	L	H	Short		<bits 383–368>	<bits 367–352>	Write seg 1 to CR
18	H	L	H	Short		<bits 415–400>	<bits 399–384>	Write seg 2 to CR
19	H	L	L	Long		<bits 447–432>	<bits 431–416>	Write seg 3 to CR
20	L	H	H	Medium		*SR[31:16]	*SR[15:0]	Read Status reg
If a match was not found then the 512-bit data word was not found, therefore the search may end. If a match was found then continue.								
21	H	L	H	Short		0000H	ssssH	Write seg 0 to CR (with TAG = ssssH: see note)
22	H	L	H	Short		0000H	0000H	Write seg 1 to CR
23	H	L	H	Short		<bits 479–464>	<bits 463–448>	Write seg 2 to CR
24	H	L	L	Long		<bits 511–496>	<bits 495–480>	Write seg 3 to CR
25	L	H	H	Medium		*SR[31:16]	*SR[15:0]	Read Status reg

Notes:

1. *SR[31:16] , *SR[15:0] = If there was a match during the comparison, the contents of the Status register are valid and appear on DQ[31:0]. If no-match occurs, all the devices are in the tri-state condition to prevent bus contention. Pull-up resistors are required to drive the bus to 0xFFFFFFFF on a no-match condition.
2. ppppH = The TAG bits for this CAM word. It is the address where the first CAM word was located.
3. qqqqH = The TAG bits for this CAM word. It is the address where the second CAM word was located.
4. rrrrH = The TAG bits for this CAM word. It is the address where the third CAM word was located.
5. ssssH = The TAG bits for this CAM word. It is the address where the fourth CAM word was located.

Table 4: CAM Cycle Sequence for Searching for an Entry

```
bool search(unsigned long data_array[4][5])        /*data_array holds the 512-bit data word. It
                                                   is five CAM words and segment 0 of each word
                                                   has 0000H where the TAG will be placed */
{
unsigned long tag, status;
unsigned int location;

tag = 0x0000FFFE; /* the first TAG is always FFFE */
for ( location = 0; location < 5; location++)
   {
   data_array[0][location] |= tag;                  /* add the TAG to segment 0 of each CAM word */
    dws(data_array[0][location]);                   /* data write short - first segment */
    dws(data_array[1][location]);                   /* data write short - second segment */
    dws(data_array[2][location]);                  /* data write short - third segment */
    dwlec(data_array[3][location]);                /* data write long with /ec low - fourth segment */
    status = crm();                                /* command read medium to read status register */
    if ( (status & 0x80000000) == 0 )              /* match found in memory array */
                                                   /*(requires pull-up resistor on DQ31)*/
      {
       tag = status & 0x0000FFFF;                  /* the tag for the next CAM word is the
                                                   address of the highest-priority match */
      }
     else
      {
      return FALSE;                                /* the routine has found that the 512-bit word is not in the CAM */
      }                          /* please note that FALSE and TRUE must be assigned values using #define*/
     }/* end for */
 return TRUE;                                      /* the routine has found that the 512-bit word is in the CAM */
}/* end search */
```

Figure 5: C Code for Searching the CAM for an Entry

Deleting Data Words from the CAM

Data words are added to the CAM in a way that will allow CAM words to be shared among entries. For this to occur the data "slice" and the associated TAG from one entry must be identical to another. This usually occurs when entries differ only in the end CAM words. If the first CAM words are identical then they can be shared. It is impossible for only the last CAM words of an entry to be shared. Figure 6 shows how 512-bit entries would have been located in the CAM memory array if some CAM word sharing was possible during addition.

It can be seen that some of the CAM words form part of more than one 512-bit entry. This means that when deleting entries, care must be taken not to delete individual CAM words that are part of more than one entry. For example, the first three CAM words in Figure 6 can not be deleted at present because they form part of three separate 512-bit entries. The TAG of each CAM word must be tested to check that the CAM word that they point to is not part of multiple 512-bit entries. The test is completed in reverse working from TAG 5 through TAG 2. The start TAG = FFFFh, does not need to be tested as it does not signify an address of a linked CAM word. When a multiple entry condition is found it can be assumed that the subsequent CAM words also form part of multiple entries and need not be tested. It is also known that the fifth CAM word of any 512-bit entry is always part of that entry only and can be safely deleted.

The routine in Table 5 on page 10 shows how a 512-bit data word can be removed from a 128-bit wide CAM. The software implementation of this routine, as shown in Figure 7 on page 13, simplifies using traditional looping methods. The 512-bit word will be located in the CAM memory array split up into five 128-bit CAM words. The 512-bit data word to be removed from the CAM must be located using the same procedure as the search routine. This will find the five CAM words and their associated TAGs. The TAGs that were given when they were added to CAM will be located in Segment 0 of each CAM word. The five TAGs will be known as T1, T2, T3, T4, and T5. T1 is the first TAG and is always FFFEH.

The Delete routine is split into two parts. The first part, which is the same as the search routine, is described in lines 1 through 25. The second part, which deletes the individual CAM words, is described in lines 26 through 48.

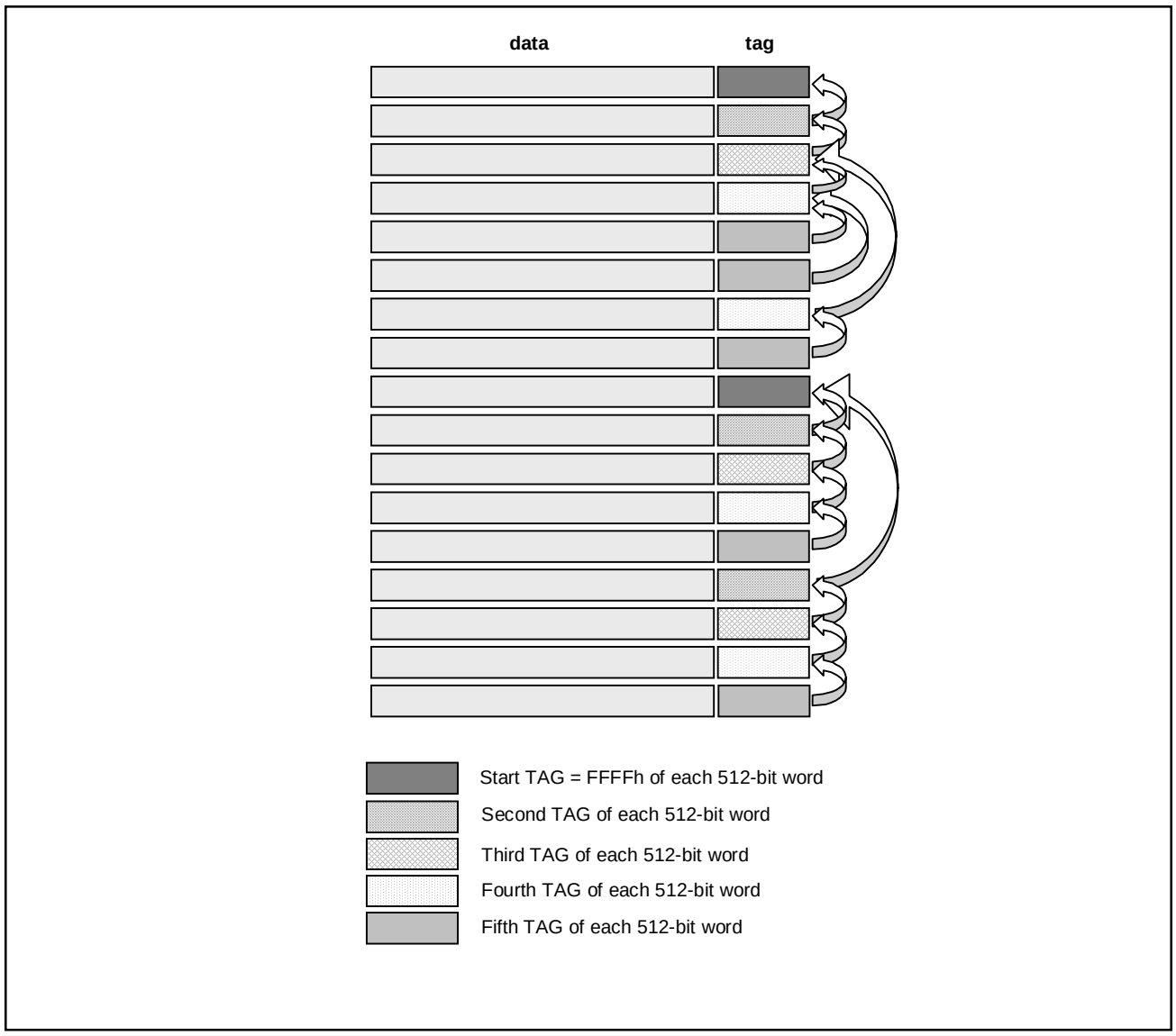


Figure 6: CAM Words Being Shared by 512-bit Entries

Lines 1 through 4 write the first of the five CAM words to the Comparand register. This word has been given the start TAG FFFEH (T1). An automatic compare is done after the last segment is written. A match can be determined from the /MF pin of the last device in the chain, or by looking at DQ31 in the Status register during the Status register read on line 5. DQ31 will be driven LOW by the highest-priority matching device, or it will be pulled HIGH by a pull-up resistor that must be added to DQ31 when using this method. At this point a decision has to be made, based on whether a match exists or not.

(a) If there is no-match, the 512-bit word to be removed cannot be found in the CAM. There is no need to compare the remaining CAM words. The routine ends here. The word cannot be deleted, as it is not located in the CAM array.

(b) If there was a match after the automatic compare, the TAG for the next CAM word is found by masking the contents of the Status register with 0000FFFFH. This TAG is used in line 6 and will be known as T2.

This is repeated until all the CAM words have been searched for in the CAM. When the last CAM word compare has given a match, the five TAGs have been found (T1, T2, T3, T4, T5). They can now be used to remove the 512-bit word from the CAM.

Line	/CM	/W	/EC	Cycle Length	Mnemonic	DQ[31:16]	DQ[15:0]	Description
1	H	L	H	Short		<bits15–0>	FFFEH(T1)	Write seg 0 to CR (with start TAG = FFFEH)
2	H	L	H	Short		<bits 47–32>	<bits 31–16>	Write seg 1 to CR
3	H	L	H	Short		<bits 79–64>	<bits 63–48>	Write seg 2 to CR
4	H	L	L	Long		<bits 111–96>	<bits 95–80>	Write seg 3 to CR
5	L	H	H	Medium		*SR[31:16]	*SR[15:0]	Read Status reg
If a match was not found then the 512-bit data word was not found, therefore the routine will end because it will be unable to delete the word. If a match was found then continue.								
6	H	L	H	Short		<bits 127–112>	T2	Write seg 0 to CR (with TAG = T2: see note)
7	H	L	H	Short		<bits 159–144>	<bits 143–128>	Write seg 1 to CR
8	H	L	H	Short		<bits 191–176>	<bits 175–160>	Write seg 2 to CR
9	H	L	L	Long		<bits 223–208>	<bits 207–192>	Write seg 3 to CR
10	L	H	H	Medium		*SR[31:16]	*SR[15:0]	Read Status reg
If a match was not found then the 512-bit data word was not found, therefore the routine will end because it will be unable to delete the word. If a match was found then continue.								
11	H	L	H	Short		<bits 239–224>	T3	Write seg 0 to CR (with TAG = T3: see note)
12	H	L	H	Short		<bits 271–256>	<bits 255–240>	Write seg 1 to CR
13	H	L	H	Short		<bits 303–288>	<bits 287–272>	Write seg 2 to CR
14	H	L	L	Long		<bits 335–320>	<bits 319–304>	Write seg 3 to CR
15	L	H	H	Medium		*SR[31:16]	*SR[15:0]	Read Status reg
If a match was not found then the 512-bit data word was not found, therefore the routine will end because it will be unable to delete the word. If a match was found then continue.								
16	H	L	H	Short		<bits 351–336>	T4	Write seg 0 to CR (with TAG = T4: see note)
17	H	L	H	Short		<bits 383–368>	<bits 367–352>	Write seg 1 to CR
18	H	L	H	Short		<bits 415–400>	<bits 399–384>	Write seg 2 to CR
19	H	L	L	Long		<bits 447–432>	<bits 431–416>	Write seg 3 to CR
20	L	H	H	Medium		*SR[31:16]	*SR[15:0]	Read Status reg
If a match was not found then the 512-bit data word was not found, therefore the routine will end because it will be unable to delete the word. If a match was found then continue.								
21	H	L	H	Short		0000H	T5	Write seg 0 to CR (with TAG = T5: see note)
22	H	L	H	Short		0000H	0000H	Write seg 1 to CR
23	H	L	H	Short		<bits 479–464>	<bits 463–448>	Write seg 2 to CR
24	H	L	L	Long		<bits 511–496>	<bits 495–480>	Write seg 3 to CR
25	L	H	H	Medium		*SR[31:16]	*SR[15:0]	Read Status reg

Table 5: CAM Cycle Sequence for Deleting an Entry

After the search has been completed, all the TAGs are now known. If at any time the automatic compare caused a no-match condition, the 512-bit word was not found in the CAM. Therefore the delete routine will fail, as it will not be able to proceed. If all the TAGs were located, the routine will proceed by removing the five CAM words one at a time.

The five individual CAM words are stored in the CAM memory array linked together by their TAGs. Each word is linked to the preceding word in a chain. The first has the TAG FFFEH and the final word has no other CAM words linked to it.

As described earlier, CAM words can be part of more than one 512-bit entry. The fifth CAM word can be safely deleted as it will only be part of the entry being deleted. Moving through the chain from fourth word to first word, each word must be checked before it is deleted.

To check each CAM word, a comparison is done on only the TAG bits of the previous word using MR1. This shows if any other 512-bit words share the CAM word referred to by the TAG. If the comparison produces a match condition, the CAM word associated with the TAG cannot be deleted.

Line 26 deletes the last of the five CAM words by setting the validity of the highest-priority match to empty. If the search procedure produced a match condition on line 25, the highest-priority match should be the fifth CAM word. As mentioned earlier, this word can be safely deleted but the remaining CAM words must be checked before deletion. Line 27 selects the Background Register set to enable any comparisons through a Mask register. Line 28 writes Segment 0 of the fifth CAM word into the Comparand register to test the TAG = T5.

Writing Segment 0 causes an automatic comparison to be performed. At this point, the Mask register will be automatically invoked to cause the comparison to be performed on only the 16 TAG bits. A match can be determined from the /MF pin of the last device in the chain, or by looking at DQ31 in the Status register during the Status register read on line 29. DQ31 will be driven LOW by the highest-priority matching device, or will be pulled HIGH by a pull-up resistor that must be added to DQ31 when using this method. At this point a decision must be made, based on whether a match exists or not:

(a) If there was a match, the 128-bit CAM word specified by the TAG is part of more than one 512-bit word. Therefore the next and subsequent CAM words cannot be removed from the memory array. The Delete routine can stop at this point as no other CAM words can be deleted as they form more than one 512-bit entry.

(b) If there was a no-match after the automatic compare, the TAG was not found anywhere else in the CAM memory array. The next CAM word can be removed. Line 30 selects the appropriate device where the 128-bit word is known to be located. Line 31 deletes the word by setting the validity of the CAM word specified by T5 to empty. Line 32 selects all the devices in the daisy chain to respond to subsequent instructions.

The remaining TAGs (T4, T3, and T2) are tested in the same way. T1 (FFFEH) does not need to be tested as this does not represent an actual CAM word address. If the routine encounters a match condition after any of the automatic comparisons, it will skip to line 48. This selects the Foreground Register set to return the system to normal operating mode.

Line	/CM	/W	/EC	Cycle Length	Mnemonic	DQ[31:16]	DQ[15:0]	Description
26	L	L	H	Long	VBC_HM_E	042DH	xxxxH	Delete CAM word 5 by setting the validity to empty
27	L	L	H	Short	SBR	0619H	xxxxH	Select Background Register set
28	H	L	L	Long		0000H	T5	Write seg 0 to CR (with TAG = T5: see note)
29	L	H	H	Medium		*SR[31:16]	*SR[15:0]	Read Status reg
If no-match: continue. If there is a match, the deletion routine may skip to line 48 as subsequent CAM words cannot be deleted as they are part of more than one 512-bit entry.								
30	L	L	H	Short	TCO_DS	0A28H	T5/1024	Select appropriate device : see note
31	L	L	H	Long	VBC_aaaH_E	0C25H	T5	Delete CAM word 4 by setting the validity to empty
32	L	L	H	Short	TCO_DS	0A28H	FFFFH	Select all devices
33	H	L	L	Long		<bits 351-336>	T4	Write seg 0 to CR (with TAG = T4 : see note)
34	L	H	H	Medium		*SR[31:16]	*SR[15:0]	Read Status reg

Table 5: CAM Cycle Sequence for Deleting an Entry (*continued*)

If no-match: continue. If there is a match, the deletion routine may skip to line 48 as subsequent CAM words cannot be deleted as they are part of more than one 512-bit entry.

Line	/CM	/W	/EC	Cycle Length	Mnemonic	DQ[31:16]	DQ[15:0]	Description
35	L	L	H	Short	TCO_DS	0A28H	T4/1024	Select appropriate device : see note
36	L	L	H	Long	VBC_aaaH_E	0C25H	T4	Delete CAM word 3 by setting the validity to empty
37	L	L	H	Short	TCO_DS	0A28H	FFFFH	Select all devices
38	H	L	L	Long		<bits 239-224>	T3	Write seg 0 to CR (with TAG = T3 : see note)
39	L	H	H	Medium		*SR[31:16]	*SR[15:0]	Read Status reg

If no-match: continue. If there is a match, the deletion routine may skip to line 48 as subsequent CAM words cannot be deleted as they are part of more than one 512-bit entry.

40	L	L	H	Short	TCO_DS	0A28H	T3/1024	Select appropriate device : see note
41	L	L	H	Long	VBC_aaaH_E	0C25H	T3	Delete CAM word 2 by setting the validity to empty
42	L	L	H	Short	TCO_DS	0A28H	FFFFH	Select all devices
43	H	L	H	Short		<bits 127-112>	T2	Write seg 0 to CR (with TAG = T2 : see note)
44	L	H	H	Medium		*SR[31:16]	*SR[15:0]	Read Status reg

If no-match: continue. If there is a match, the deletion routine may skip to line 48 as subsequent CAM words cannot be deleted as they are part of more than one 512-bit entry.

45	L	L	H	Short	TCO_DS	0A28H	T2/1024	Select appropriate device : see note
46	L	L	H	Long	VBC_aaaH_E	0C25H	T2	Delete CAM word 1 by setting the validity to empty
47	L	L	H	Short	TCO_DS	0A28H	FFFFH	Select all devices
48	L	L	H	Short	SFR	0618H	xxxxH	Select Foreground Register set

Notes:

1. *SR[31:16] , *SR[15:0] = If there was a match during the comparison, the contents of the Status register are valid and appear on DQ[31–0]. If no-match occurs, all the devices are in the tri-state condition to prevent bus contention. Pull-up resistors are required to drive the bus to 0xFFFFFFFF on a no-match condition.
2. Tn/1024 = The appropriate device must be selected using a Temporary Command Over-ride instruction. The TAG indicates the address of the CAM word to be deleted. Bits 15-10 of the TAG indicate the page address (and therefore the appropriate device) of the CAM word. In this case, the correct device is given by dividing the TAG by 1024, but similar solutions are possible.
3. T2 = The TAG bits for the second CAM word. It is the address where the first CAM word was located.
4. T3 = The TAG bits for the third CAM word. It is the address where the second CAM word was located.
5. T4 = The TAG bits for the fourth CAM word. It is the address where the third CAM word was located.
6. T5 = The TAG bits for the fifth CAM word. It is the address where the fourth CAM word was located.

Table 5: CAM Cycle Sequence for Deleting an Entry (*continued*)

```
void delete(unsigned long data_array[4][5])                    /* data_array holds the 512-bit data word. It
                                                                  is five CAM words and segment 0 of each word
                                                                  has 0000H where the TAG will be placed */
{
unsigned long status, tag;
unsigned int location;

if (search(data_array) == TRUE)
  /* this uses the search routine to locate all the TAGs and affixes them to the CAM words */
{
 cwl(0x042D0000);                    /* VBC_HM_E - the last CAM word can be removed as it has no other CAM
                                                                                    words linked to it. */
 cws(0x06190000);                  /* command write short - set background register set */
 for ( location = 5; location > 1; location—) /* work back from CAM word 5 to CAM word 2 */
   {
    dwlec(data_array[0][location]);            /* data write long with /ec low - segment 0.
                                                                                    This is to do a compare on the TAG */
    status = crm();                                          /* command read medium - read status register */
     if ( (status & 0x80000000) != 0 )                                  /* no match found in memory array */
      {                                                                 /* (requires pull-up resistor on DQ31 */
      tag = data_array[0][location] & 0x0000FFFF;                          /*separate tag from segment 0 */
      cws(0x0A280000 & tag/1024);                                      /* TCO_DS select correct device */
      cwl(0x0C250000 & tag);                                 /* VBC_aaa_E: delete CAM word pointed to by the TAG */
      cws(0x0A28FFFF);                                                 /* TCO_DS 0xFFFF select all devices */
      }
      else
      {
      break;                                        /* there was not a match on the TAG comparison, therefore no more
                                                                                    of the CAM words can be deleted */
      }/* end if */
    }/* end for */
 cws(0x06180000);                                                /* command write short – set foreground register set */
}/* end if */
}/* end delete */
```

Figure 7: C Code for Deleting a CAM Entry

TIME CRITICAL APPLICATIONS

It can be seen that the procedure to delete an entry from the CAM is very time consuming. This is due to the need to: (a) locate the TAGs, (b) test the individual CAM words to make sure they are not part of other entries, and finally (c) delete the appropriate 128-bit words. In time critical applications, the deletion routine may cause congestion if more important operations have to wait until a deletion is finished. It is usually true that searches are more important than deletions and therefore have a higher priority. In this case, the deletion can be delayed if the more important search has to be performed.

If there are no searches pending, deletions may be done in complete procedures. If searches are required once a deletion has started, it is possible to interrupt the deletion to perform the more important tasks. Once these tasks have been cleared, the deletion can be returned to and completed. The deletion can be split up into five smaller routines that must be completed in the order specified:

1. Search the CAM to locate the individual TAGs of each CAM word and delete the fifth CAM word.
2. Test the fourth CAM word to make sure it is not part of more than one entry and delete if possible.
3. Test the third CAM word and delete if possible.
4. Test the second CAM word and delete if possible.
5. Test the first CAM word and delete if possible.

Figure 8 shows how the operations can be interleaved if searches need to be done once a deletion has begun.

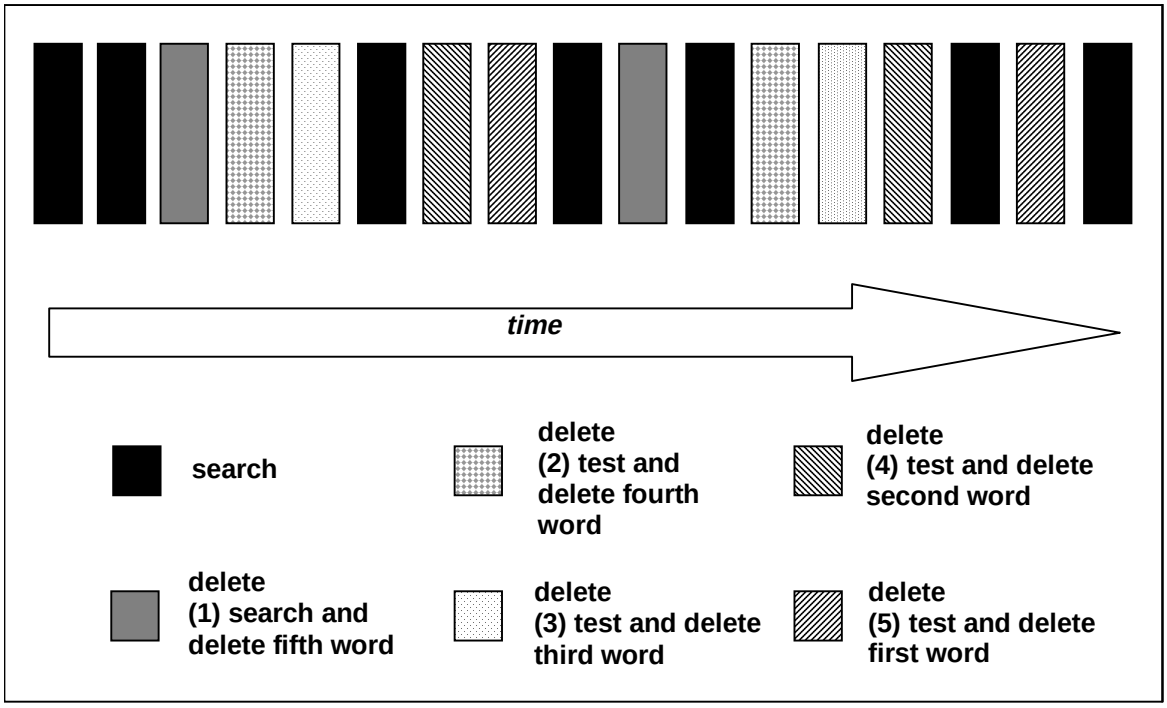
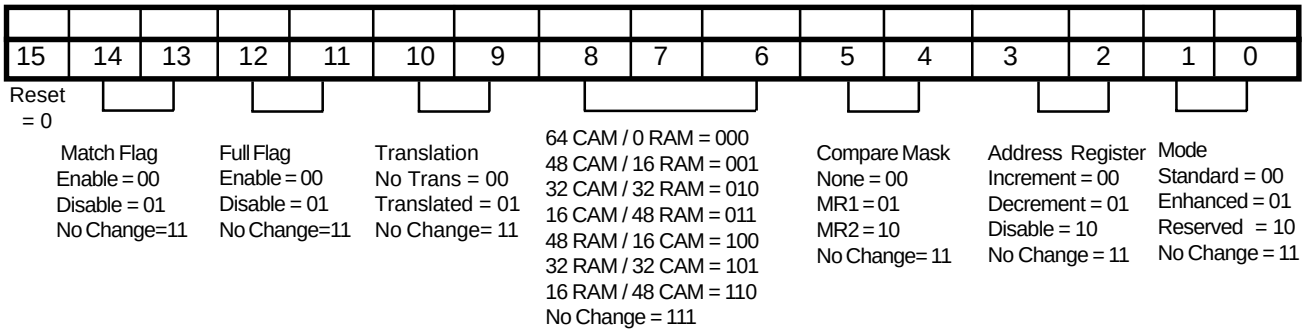


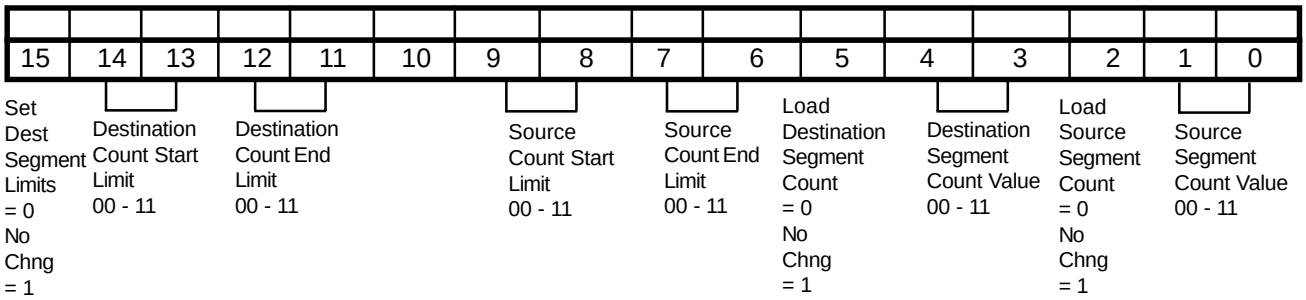
Figure 8: Interleaving Searches with Deletions

	/CM	/W	/EC	Cycle Length	Mnemonic	DQ[31:16]	DQ[15:0]	Description
1								
2								
3								
4								
5								
6								
7								
8								
9								
10								
11								
12								
13								
14								
15								
16								
17								
18								
19								
20								

Blank Routine Worksheet



Control Register Bit Assignments



Segment Control Register Bit Assignments

NOTES

MUSIC Semiconductors Agent or Distributor:

MUSIC SEMICONDUCTORS

http://www.music-ic.com
email: info@music-ic.com

USA Headquarters
MUSIC Semiconductors
254 B Mountain Avenue
Hackettstown, New Jersey 07840
USA
Tel: 908/979-1010
Fax: 908/979-1035
USA Only: 800/933-1550 Tech. Support
888/226-6874 Product Info.

Asian Headquarters
MUSIC Semiconductors
Special Export Processing Zone 1
Carmelray Industrial Park
Canlubang, Calamba, Laguna
Philippines
Tel: +63 49 549 1480
Fax: +63 49 549 1023
Sales Tel/Fax: +632 723 62 15

European Headquarters
MUSIC Semiconductors
Torenstraat 28
6471 JX Eygelshoven
Netherlands
Tel: +31 45 5462177
Fax: +31 45 5463663