SECTION 1

THE MAPPER AND US

1.1 WHAT IS THE MAPPER?

The Mapper is a chip that is used on the 99/8 to generate 24 address lines from the 16 TMS9995 address lines. It does this by using the Most Significant (MS) four 9995 address lines to select one of sixteen different 32-bit registers located within the Mapper chip, and then adding the Least Significant (LS) twelve bits of the 9995 address bus to the 24-bit address portion of the selected register to generate the full 24-bit address. These 32-bit registers are termed base registers. This scheme allows one to "map to a byte resolution"; that is, any byte in the 9995 address space can map to any byte in the 24-bit address space.

As the MS 4-bits of the LAS are used to select one of sixteen different 24-bit base registers, each base register handles the base of a 4K byte space in the 24-bit side of the mapper. This may be easier to see if we look at it from the point of view that the LS twelve 9995 address lines are added to the base register, and twelve address lines will handle a 4K space.

The 16-bit 9995 address space is termed the Logical Address Space (LAS), and the 24-bit address space is termed the Physical Address Space (PAS).

The 99/8 has some memory oriented devices including both SRAM and ROM that are designed to respond in the LAS, and the mapper is fed that condition to inhibit the mapping action. Note that this means that when you access one of these dedicated LAS locations no mapping will occur. See the LAS memory map for the 99/8 for more details.

Also, the 99/8 has some internal memory oriented devices that respond in the PAS. Included are the internal 64K byte DRAM (based at >000000), the Command Module Port memory space (16K based at >FF6000), a 16K byte ROM based at >FFA000, and two DSR ROM spaces based at >FF4000. Mapping occurs for these blocks, but data is available only from the 99/8 console (you can't get none from an expansion unit). See the LAS memory map for the 99/8 for more details.

1.2 WHERE THE MAPPER REGISTER IMAGES ARE LOCATED

The Mapper Register images are located in the SRAM, and start with the Most Significant Byte (MSBY) in the very first byte in the SRAM. Since the SRAM may be located in one of two different locations (>8000 or >F000 based), we will "zero base" the following layout. There are eight different 64 byte Mapper Register images possible. The other seven 64 byte images are piled directly on top of the one zero based. To get the actual SRAM location, simply put either an 8 or a F in front of the HEX SRAM location shown below.

MS 4 9995 Addr Bits	SRAM Location In HEX	Function
0000	000	MSBY of Mapper Reg 0
0000	001	Next to MSBY of Reg 0
0000	002	Next to LSBY of Reg 0
0000	003	LSBY of Mapper Reg 0
0001	004	MSBY of Mapper Reg 1
0001	005	Next to MSBY of Reg 1
0001	006	Next to LSBY of Reg 1
0001	007	LSBY of Mapper Reg 1
0010	008	MSBY of Mapper Reg 2
0010	009	Next to MSBY of Reg 2
0010	00A	Next to LSBY of Reg 2
0010	00B	LSBY of Mapper Reg 2
0011	00C	MSBY of Mapper Reg 3
0011	00D	Next to MSBY of Reg 3
0011	00E	Next to LSBY of Reg 3
0011	00F	LSBY of Mapper Reg 3
0100	010	MSBY of Mapper Reg 4
0100	011	Next to MSBY of Reg 4
0100	012	Next to LSBY of Reg 4
0100	013	LSBY of Mapper Reg 4
0101	014	MSBY of Mapper Reg 5
0101	015	Next to MSBY of Reg 5
0101	016	Next to LSBY of Reg 5
0101	017	LSBY of Mapper Reg 5

MS 4 9995 Addr Bits	SRAM Location In HEX	Function
0110	018	MSBY of Mapper Reg 6
0110	019	Next to MSBY of Reg 6
0110	01A	Next to LSBY of Reg 6
0110	01B	LSBY of Mapper Reg 6
0111	01C	MSBY of Mapper Reg 7
0111	01D	Next to MSBY of Reg 7
0111	01E	Next to LSBY of Reg 7
0111	01F	LSBY of Mapper Reg 7
1000	020	MSBY of Mapper Reg 8
1000	021	Next to MSBY of Reg 8
1000	022	Next to LSBY of Reg 8
1000	023	LSBY of Mapper Reg 8
1001	024	MSBY of Mapper Reg 9
1001	025	Next to MSBY of Reg 9
1001	026	Next to LSBY of Reg 9
1001	027	LSBY of Mapper Reg 9
1010	028	MSBY of Mapper Reg 10
1010	029	Next to MSBY of Reg 10
1010	02A	Next to LSBY of Reg 10
1010	02B	LSBY of Mapper Reg 10
1011	02C	MSBY of Mapper Reg 11
1011	02D	Next to MSBY of Reg 11
1011	02E	Next to LSBY of Reg 11
1011	02F	LSBY of Mapper Reg 11
1100	030	MSBY of Mapper Reg 12
1100	031	Next to MSBY of Reg 12
1100	032	Next to LSBY of Reg 12
1100	033	LSBY of Mapper Reg 12
1101	034	MSBY of Mapper Reg 13
1101	035	Next to MSBY of Reg 13
1101	036	Next to LSBY of Reg 13
1101	037	LSBY of Mapper Reg 13

MS 4 9995 Addr Bits	SRAM Location In HEX	Function
1110	038	MSBY of Mapper Reg 14
1110	039	Next to MSBY of Reg 14
1110	03A	Next to LSBY of Reg 14
1110	03B	LSBY of Mapper Reg 14
1111	03C	MSBY of Mapper Reg 15
1111	03D	Next to MSBY of Reg 15
1111	03E	Next to LSBY of Reg 15
1111	03F	LSBY of Mapper Reg 15

1.3 HOW TO USE THE MAPPER

Before any mapping action can occur, we must first set up the Mapper; ie, we must first generate in the SRAM the IMAGE (duplicate) of the Mapper Registers, and then cause this image to be copied into the Mapper chip.

We can generate the image for the Mapper Chip by hand entering through JT BUG all 64 bytes. It is a good idea to enter all whether you need'em or not. This philosophy may well prevent strange things from happening. If there was enough space left in JT BUG, perhaps the author could add some data and a transfer command. The following program can be used to load that area with all zeros.

```
8400   0200  LI   R0,>0000      SET UP TO LOAD ZEROS
8402   0000
8404   0201  LI   R1,>8000      SET UP STORE START ADDRESS
8406   8000
8408   CC40  MOV  R0,*R1+       MOVE WORD TO MAPPER
840A   0281  CI   R1,>8400      SEE IF THROUGH
840C   8040
840E   11FC  JLT  -4            MOVE AGAIN IF NOT THRU
8410   0420  BLWP 0             RETURN TO JT BUG
8412   0000
```

Once this data base is generated, we then must cause it to be written into the control register.

1.3.1 The Control Register. The Control Register is used to initiate either a Mapper Register load or save. the format is as follows

!0000 CCCL! where CCC selects one out of eight successive image locations in the SRAM, and L determines a LOAD or SAVE operation.

!0000 0001! causes the Mapper Registers to be loaded from the base location in the SRAM.

!0000 0010! causes the Mapper Registers to be saved at >40 from the base location in the SRAM.

A MOVB to the Mapper location caused the LOAD or SAVE operation to be executed.

The Control Register is based at >8810 on power-up, and may be moved to >F870 by a CRU SBZ operation. See the 99/8 specs for more information.

1.3.2 The Status Register. The Status Register is used to sense the state of the protect violation bits. A MOVB from the Status Register causes the bit(s if two or three happen to be set) to be reset (the interrupt condition to be cleared). The Status Register is located at the same address as that for the Control Register.

1.4 HOW TO USE THE PROTECT BITS

There are three protect bits associated with each 24-bit base register, and if violated, an interrupt to the uP occurs. The MSB in the 32-bit field associated with each base register is a WRITE protect bit. It actually inhibits a WRITE operation to the area that is protected, and the full 4K byte block is protected. It cannot be less than 4K unless it is zero bytes.

The second bit into the MS byte is the EXECUTE protect bit, and its protect function is to create an interrupt if the uP gets an instruction from a protected area. It does NOT try to inhibit the execution of the instruction fetched from the protected area.

The third bit into the MS byte is the READ protect bit, and a violation occurs on a READ access that is not an instruction fetch READ. If protected, the READ operation occurs normally, and the interrupt output of the Mapper is set LOW.

A full 4K byte LAS memory block is protected if any of the protect bit is set to a "1".

1.5 THE SRAM CHIP SELECT

The SRAM is accessed at >8000 in the LAS on power-up, and may be moved to >F000 by a CRU SBZ operation. See the 99/8 specs for more information.

1.6 THE DRAM CHIP SELECT

The DRAM is based at >000000 in the PAS, and the Mapper must be set up to point to it before it may be accessed.

1.7 EXAMPLES

It is desired to map the 4K byte LAS block based at >A000 into the onboard DRAM, but displaced into it by >1000 (the DRAM has a span from >000000 to >00FFFF). I do not wish to use any of the three protect bits as I would have to write an interrupt processor routine, also. I intend that every time I execute a MOVB @>A000,R0, I get a byte from >001000 in the PAS, and put it into R0.

First, I am going to use JT BUG to zero out the base register set located at the base of the SRAM (>8000 or >F000). Then I will change the locations of base register 10 (>A) to be as follows.

SRAM Address	Data	
028	0000 0000	no protect bits are on
029	0000 0000	MSBY in the PAS is >00
02A	0001 0000	mid byte in the PAS is >10
02B	0000 0000	LSBY in the PAS is >00

Now, when we execute the MOVB instruction, the mapper will form the PAS by accessing the #10 base register to get >001000, and then adding >000 (this value comes from the >A000 in the MOVB instruction with the "A" left off).

```
    001000   from the #10 Mapper base register
  +    000   from the MOVB instruction
  --------
    001000   for the PAS effective address
```

If we did a MOVB >A5FC,R0, then the mapper will function as follows.

```
    001000   from the #10 Mapper base register
  +   5FC    from the MOVB instruction
    ------
    0015FC   for the PAS effective address
```

Assuming that we had loaded the mapper as shown, executed a MOVB @>AF52,R0, then the mapping will be as follows.

SRAM Address	Data	
028	0000 0000	no protect bits are on
029	0000 0000	MSBY in the PAS is >00
02A	0011 0110	mid byte in the PAS is >36
02B	0010 0111	LSBY in the PAS is >27

```
    003627   from the #10 Mapper base register
  +   F52    from the MOVB instruction
    ------
    004579   for the PAS effective address
```

1.8 ADDITIONAL MAPPER FUNCTIONS

In addition to mapping duties, the Mapper chip is responsible for generating memory control signals very similar to those of the TMS9900 uP chip. The Mapper also generates a 10.7MHz/33 clock for the 99/8 Speech Synthesizer.

GOOD LUCK.