

FT-68X MMU DESCRIPTION

James R. Schmidt

FORWARD TECHNOLOGY INC.
San Jose, CA

This specification is on the memory management unit for the FT-68X series of Forward Technology processors. First we begin by defining some terms.

All addresses are given in hexadecimal. We will either write the word hex, precede the numeric digits with "0x", or both. These forms are synonymous.

The Virtual Address is the address 24 bits wide coming out of the 68000 processor chip and being presented to the memory translation hardware; i.e. the segment and page map registers. The logical address occurs in the context space as given by the context register which is independently writable by the programmer at address hex E0000. Sixteen contexts are possible, labeled zero through fifteen. They are written by storing a context of positive polarity in the context register at 0xE0000. The bits for the context must be stored in the high order four bits of the data word. For example, to write context seven into the context register, one would load a 68000 register with 7000 hex and write it to location E0000. The virtual address so far untranslated, is the address generated by the software program being run. All virtual addresses come from the 68000 chip. All addresses less than 0x200000 go completely through the memory management unit. Alternately stated, if the virtual address resides in the lower 21 bits of the processor's 24 bit address range, it is translated by the memory management hardware. Addresses above 0x200000 (hex) refer to local, on-board I/O space and are given in the 68X User's Manual. Any address which we have herein called a virtual address, lying between hex locations 0 and hex location 1FFFFFF, that is one less than hex 200000, are translated by the memory management hardware. The memory management hardware consists of the segment map and the page map. There are two levels of translation. The first level of translation occurs in the segment map where the virtual address is translated into a logical address to later be presented to the page map for further translation. The high order six bits of the virtual address are translated in the segment map to produce an eight bit output from the segment map. To be more precise, the four bits from the context

register plus address bits A15 - A20, inclusive, are fed into the segment map. These ten bits address one of 1024 locations, the size of the segment map. The output is eight bits of logical address presented to the page maps, and four bits of protection information. This segment protection information is fed to a memory error prom whose function is to catch addressing violations and produce a bus error. Now, the eight address bits which on the schematic are called XA15 through XA22, are fed directly to the page map. The high order two bits, XA21 and XA22, are not used in the current implementation. XA15 through XA20 and the virtual address bits A11 through A14 form 10 bits of address supplied to the page map. These ten bits of address, again virtual address bits A11 through A14, and logical address bits XA15 through XA20, select one of 1024 page map registers. The output of the page map is the physical memory address, that is the address that is actually passed to the physical memory, whether it resides on the processor board, a multibus memory board, a multibus I/O space board, or the proprietary FT-768. Addresses to any of these four are output from the page map. There are twelve bits of physical address output from the page map, on the schematics labeled MA11 through MA22. Address bit numbers are always given in decimal on the schematic. The low order eleven virtual address bits combine with the high order twelve page map output address bits to form the 23 bit physical address that is applied to the physical memory, whether it be local memory, extended memory, multibus memory or multibus I/O as previously described. The high order four bits output from the page map are used for mode control. Recall now that there are sixteen total bits output from the page map, while there are only twelve total bits output from the segment map. The page map high order four bits are taken as follows.

The most significant bit is labeled the "used" bit. This bit is set automatically by the processor hardware whenever any location within that page is accessed.

The second most significant bit is the dirty bit. This bit is set whenever any location within the page is written, but not when it is read. Together these two bits allow the operating system to keep track of dirty pages, that is modified pages, for later swapping or for virtual memory management.

The next two bits which would be identified as the 13th and 12th bits are taken together as a unit. If they have value binary 00 we are talking about a local memory access. Value one, that is binary 01, gives a local I/O address. Local I/O is not implemented. Therefore, this gives an effective non-existent memory location. This is a valuable mode for operating system designers which allows them to mark unused pages of a process's address space non-existent,

which means that an attempted access here by a malicious or erroneous user program, would cause a bus error and be trapped without harm by the operating system. These two bits, with value binary 10, translate to bus memory, or more precisely, multibus memory. The final condition, that is value binary 11, is bus I/O, or more precisely, multibus I/O. These two bits, as you can see, give you the opportunity to break your addressing range into four address spaces, local memory address space, non-existent address space, multibus memory address space and multibus I/O address space.

Now, let's have a brief summary. The context register is a four bit register that can be written to, using the high order four bits of the data bus, by writing to local on-board location hex E0000. The segment maps are a 1024 register map which can be written to using the low order twelve bits of the data bus with the address to be written into the segment maps forming the low order eight bits, D0 through D7, and the protection bits in the next four bits, that is D8 through D11 inclusive. The segment map is written at the address it is also read, that is the first segment map register resides at 0xCC0000. The second segment map register resides at 0xC08000. The third segment map register resides at 0xC10000, the fourth at 0xC18000. As you can see, each register is written with an address delta hex 8000. Hex 8000 is 32768, or precisely the size in bytes of a segment. We have here then, that addressing the segment registers to modify their contents is done by using the same address that would read out their contents, under normal program flow. This makes it quite straightforward, conceptually, to keep track of which segment register you are attempting to modify. The page map registers work in a similar fashion. The page map registers are addressed after the segment map registers are set up. As mentioned earlier, every address of every program that falls between hex 0 and hex 200000 is translated. This includes addresses of instructions whose intent is to modify the segment or page maps. Therefore, a small amount of planning is required to set up the board initially, that is to write the contents of the segment maps and page maps. If a simple rule of thumb is followed, it becomes quite straightforward. The rule of thumb is, set up the segment map registers first mapped straight through, that is virtual address is identical to logical address, i.e. write hex C into segment map register C, hex 8000 into segment map register 1, which is addressed at hex 8000, hex 10000 into segment map register 2 which is addressed at hex 10000 and so on. This gives you a linear two megabyte address space for your logical address to address your page maps and will result in a very simple method of writing your entire 1024 register page map. In the page map, each page is hex 800 bytes, thus the first page map exists at location hex A00000. The second register exists at hex A00800, the third register at A01000, the fourth register at AC1800 and so on. The page map registers

are thus written with their contents at steadily increasing addresses with a delta value of hex 800 and an offset of CxA0000. The contents of the page map registers are written by putting the 12 bits of physical address in the low order 12 bits of the 16 bit word and the 4 mode bits into the high order 4 bits of the 16 bit word. Thus the page map mode control exists in the high order nibble of the 16 bit word or data lines D15 thru D12. Data lines D11 thru D0 contain the 12 most significant bits of the physical address.

Now a few examples to help clarify the issues.

Lets assume that the segment map is set up in a linear address space fashion as described earlier, that is segment register 0 contains 0, segment register 1 contains 8000, segment register 2 contains 10000 hex and so on. If we wish to write virtual address 0x0 to physical address Cx0 in local memory on the processor board we would then write a Cx0000 into the page map register 0xC. If however we wish to write to physical location Cx800 when we wrote to virtual address CxC we would write 0xC001 into the page map register CxC. If we wished at virtual address 0x0 to actually access multibus memory address Cx0 we would load the page map with Cx2000. If with virtual address 0xC we wished to write to multibus I/O space 0x0 we would load the page map with Cx3000. If, as a final example we wished to write with virtual address Cx0 to non-existent memory we would load the page map with Cx1000.