To increase the work RAM from 4KB ($6000–$6FFF) to 8KB ($6000–$7FFF) in the Family BASIC v3 ROM, the core changes involve redefining the top of memory (from $6FFF to $7FFF), relocating the 1KB background buffer used by BGGET/BGPUT (from $6C00 to $7C00 to keep it at the top of the expanded RAM), and updating the upper bound check in CLEAR (from $7000 to $8000). This maintains the original memory layout logic: program/variables/arrays grow upward from $6006, strings grow downward from the top ($7FFF), and the BG buffer occupies the upper 1KB (now $7C00–$7FFF)—but note that the BG buffer and string space still share this upper 1KB, so using BGGET will clobber strings (and vice versa), as in the original. The BGGET routine already checks that upward-growing memory hasn't encroached on the BG buffer start.

No other major assumptions about 4KB appear in the code—allocation relies on zpFRETOP ($03–$04) without explicit size tests. However, several hardcoded references must be patched to avoid errors or incomplete expansion. Below is a complete list of required patches, based on the disassembly. Addresses are CPU-mapped (e.g., in the $8000–$FFFF PRG ROM space). To apply these, use a hex editor on your fb3.nes ROM (PRG ROM starts at offset 0x10 in the .nes file; subtract 0x8000 from CPU addresses to get file offsets, e.g., $8662 → offset 0x662). Test thoroughly in an emulator like Mesen or FCEUX, as overflows past $7FFF could corrupt hardware registers.

1. Update memory top (zpFRETOP init) from $6FFF to $7FFF

- **Location**: InitVars routine ($8662)
- **Original bytes**: `A9 FF 85 03 A9 6F 85 04` (lda #$FF; sta $03; lda #$6F; sta $04)
- **Patched bytes**: `A9 FF 85 03 A9 7F 85 04` (change the second lda to #$7F)
- **Reason**: This sets the default zpFRETOP during initialization. All allocations reference this afterward.
- **Location**: memoryTop equate (symbolic, not a runtime load—only used in disassembly for reference)
- **Original**: `memoryTop .eq $6fff`
- **Patched**: `memoryTop .eq $7fff` (update if reassembling; otherwise, the above byte patch covers it)
- **Location**: CLEAR command memory-clear loop termination (two places: $CBDB–$CBDC and $CEF3)
- **Original**: Loop termination likely uses `E0 6F` or similar (cpx #$6F for high byte check during clear)
- **Patched**: Change to `E0 7F` (cpx #$7F)
- **Reason**: Ensures the clear loop covers the full expanded RAM without stopping early.
- **Location**: txtGame3 tokenized BASIC data (embedded literal bytes; find via search for $6F $FF in ROM)
- **Original bytes**: `$6F $FF`
- **Patched bytes**: `$7F $FF`
- **Reason**: Sample program data includes this address (e.g., for POKE or internal reference). Optional if not using built-in samples, but recommended.

2. Relocate bgGetRam from $6C00 to $7C00

- **Location**: bgGetRam equate (symbolic)
- **Original**: `bgGetRam .eq $6c00`
- **Patched**: `bgGetRam .eq $7c00` (update if reassembling)
- **Location**: CmdFn_BGGET memory check ($B1B8)
- **Original bytes**: `C9 6C` (cmp #$6C; checks zpVAREND+1 < $6C to ensure space for BG buffer)
- **Patched bytes**: `C9 7C` (cmp #$7C)
- **Reason**: Prevents variables from overwriting the relocated BG buffer.
- **Location**: CmdFn_BGGET pointer init ($B1CA)
- **Original bytes**: `A9 6C 85 ??` (lda #$6C; sta zpMisc+1; sets high byte of BG RAM pointer)
- **Patched bytes**: `A9 7C 85 ??` (lda #$7C)
- **Reason**: Points the buffer copy to the new location.
- **Location**: CmdFn_BGPUT pointer init ($B20B)
- **Original bytes**: `A9 6C 85 ??` (lda #$6C; sta zpMisc+1)
- **Patched bytes**: `A9 7C 85 ??` (lda #$7C)
- **Reason**: Same as above, for BGPUT restore.
- **Location**: txtGame3 tokenized BASIC data (embedded literal bytes)
- **Original bytes**: `$6C $00`
- **Patched bytes**: `$7C $00`
- **Reason**: Sample program references this address. Optional if not using samples.

3. Update CLEAR upper bound check from $7000 to $8000

- **Location**: CmdFn_CLEAR argument validation ($97D7)
- **Original bytes**: `C9 70` (cmp #$70; rejects user-supplied CLEAR addr >= $7000)
- **Patched bytes**: `C9 80` (cmp #$80)
- **Reason**: Allows CLEAR to set zpFRETOP up to $7FFF without error.
- **Location**: CLEAR command memory-clear loop termination (as in section 1; overlaps with $6FFF patches)
- **Original**: High-byte compare like `E0 70` (cpx #$70 for page $07)
- **Patched**: `E0 80` (cpx #$80)
- **Reason**: Extends the clear to cover page $07 fully.

4. Other patches for consistency

- **Location**: GAME_loadGame copy loop limit ($ADC8)
- **Original bytes**: `A9 70` (lda #$70; limits copy to before page $70)
- **Patched bytes**: `A9 80` (lda #$80)
- **Reason**: Assumes RAM ends before $7000; update to avoid truncating loads in expanded space.
- **Location**: VRAM_screen1 constant and related (used in CmdFn_BGTOOL, SaveBackgroundMap/LoadBackgroundMap, cassette headers; multiple sites)
- **Original**: Constant defined as $7000 (high byte $70 loaded via lda #>VRAM_screen1)
- **Patched**: Redefine constant to $8000 if it refers to RAM (update all lda #>$7000 to lda #$80). However, disassembly suggests this may actually be $2400 (VRAM nametable; high $24)—verify in code. If it's cassette buffer at $0700 (high $07), no change needed.
- **Reason**: Prevents mismatches if the constant is used for RAM copies or headers.
- **Location**: txtGame3 tokenized BASIC data (embedded literal bytes)
- **Original bytes**: `$70 $00`
- **Patched bytes**: `$80 $00`
- **Reason**: Sample program references this bound. Optional.

No further patches are needed—e.g., no changes to basicProgram ($6006), as it remains the bottom anchor. After these, the interpreter will treat $6000–$7FFF as usable (with the upper 1KB shared as before). If reassembling from source, update equates first. For v2.x patches in FC-DiskBASIC, details weren't explicit, but they follow similar logic (expanding from 2KB to 8KB by relocating the BG buffer and FRETOP). Test for out-of-memory errors or corruption beyond $7FFF.