Hi Turtle, I worked on this after work, I'm not done yet but these are my notes. I still need to figure out what func\_unk1 does but missing data makes it difficult. I will look at this some more after work today.

      
unk\_105C3:                    #DATA XREF: FUNC\_UNK2:off\_4B720o  
IDA disassembler named this data unk\_105c3, is called from “func\_unk2” at offset ‘4B720’, ‘o’ ordinary flow, sequential flow from one instruction to another  
        .data.b h'64        #00 hex 64  
        .data.b h'64        #01 hex 64  
        .data.b h'63        #02 hex 63  
        .data.b h'63        #03 hex 63  
        .data.b h'62        #04 hex 62  
        .data.b h'61        #05 hex 61  
        .data.b h'60        #06 hex 60  
        .data.b h'58        #07 hex 58  
        .data.b h'44        #08 hex 44  
        .data.b h'42        #09 hex 42  
        .data.b h'42        #10 hex 42  
        .data.b h'42        #11 hex 42  
        .data.b h'42        #12 hex 42  
        .data.b h'42        #13 hex 42  
        .data.b h'42        #14 hex 42  
        .data.b h'41        #15 hex 41  
        .data.b h'40        #16 hex 40  
        .data.b h'40        #17 hex 40  
        .data.b h'40        #18 hex 40  
        .data.b h'40        #19 hex 40  
        .data.b h'3F        #20 hex 3f  
  
#Function to decode:  
#r0 = h'f0 r5 = ???  
#r5 reduce r5 to a byte size  
#r5 is checked with 0xf0 and must be less than or = to 0xf0 (128 dec) to exit.  
#if r5 is less, make a copy into r3  
#shift 4 bits of r5 to the right to look at a single nibble,  
#invert 0xf0 to and with r3 so that only the bottom 4 bits are valid?  
#shift r3 to the left 12 bits  
#add r5 into r4 to pull data from table  
#r4 points to the address of the data, retrieve at 0 and 1 offset into r1 and r0  
#compare the two and branch if >=  
#else subtract r1-r0  
#back up the MACL  
#looks like this function finds the larger number and to subtract it from the smaller number and branches to do subtract the larger from the smaller if the larger number it's in a different location. to keep the value from going negative  
  
  
FUNC\_UNK1:                          #CODE XREF: FUNC\_UNK2\_00440+36p  
#IDA disassembler code named this section“func\_unk1”, code is called from “func\_unk2” with offset address of ‘00440’ + ‘36’, p is “call flow” transfer of control to a target function  
        mov.w   word\_2978E, r0        #transfer word h'f0 into r0  
        extu.b  r5, r5                #extend unsigned byte r5 with zero's from bit 8 to bit 31  
        cmp/hs  r0, r5                #compare with unsigned if r5 >= r0 set TRUE, r0 = h'f0 r5 = ???  
        bt      func\_exit1            #branch if true to loc\_29678a  
        mov     r5, r3                #transfer r5 register into r3, r3 = r5  
        shlr2   r5                    #shift right 2 bits, r5  
        shlr2   r5                    #shift right 2 bits, r5  
        not     r0, r1                #invert r0 and store in r1, Mask?  
        and     r1, r3                #r3 and r1 store in r3, Mask?  
        shll8   r3                    #shift left 8 bits r3  
        shll2   r3                    #shift left 2 bits r3  
        shll2   r3                    #shift left 2 bits r3  
        add     r5, r4                #r5 + r4, store in r4  
        mov.b   @(0,r4), r1            #transfer byte from r4 pointer + 0 byte offset into r1  
        mov.b   @(1,r4), r0            #transfer byte from r4 pointer + 1 byte offset into r0  
        extu.b  r1, r1                #extend unsigned r1 with zeros from bit8 to bit 31  
        extu.b  r0, r0                #extend unsigned r0 with zeros from bit8 to bit 31  
        cmp/hs  r1, r0                #compare if  r0 >= r1, set TRUE  
        bt      do\_math            #branch if true from above to loc\_29677a  
        mov     r1, r7                #transfer r1 into t7  
        sub     r0, r1                #r1 - r0, store in r1  
        sts     macl, r6            #store from macl into r6  
        mul.l   r3, r1                #r1 x r3, store in MAC  
        sts     macl, r1            #store from macl into r1  
        lds     r6, macl            #load to system register MACL contents of r6  
        mov     r7, r0                #transfer r7 into r0  
        shlr16  r1                    #shift right 16 bits r1  
        rts                            #return from subroutine, delayed, execute next line before returning  
        sub     r1, r0                #r0 - r1, store in r0  
  
# ---------------------------------------------------------------------------  
do\_math:                                
        sub     r1, r0                #r0 - r1, store in r0  
        sts     macl, r6            #store from macl into r6, for backup  
        mul.l   r3, r0                #r0 x r3 store in MAC  
        sts     macl, r0            #store from macl into r0  
        lds     r6, macl            #load to system register MACL contents of r6, restore from backup  
        shlr16  r0                    #shift right 16 bits r0  
        rts                            #return from subroutine, delayed, execute next line before returning  
        add     r1, r0                #r1 + r0, store in r0  
  
# ---------------------------------------------------------------------------  
func\_exit1:  
        rts                            #return from subroutine, delayed, execute next line before returning  
        mov.b   @(h'F,r4), r0        #transfer content of address contained in r4 + 0x0f into r0  
  
# End of function FUNC\_UNK1  
  
#r0 output  
#r4 input to table  
#r5 input  
#r1 + 2 input  
#r10 used for address of the subroutine call  
  
# ---------------------------------------------------------------------------  
word\_2978E:     .data.w     h'F0  
  
main:  
        mov.l   off\_4AF68, r8        #transfer long address off\_4AF68 into r8, r8 points to address FFFE8FE2  
        mov.l   off\_4B718, r1        #transfer long address off\_4B718 into r1, r1 points to address FFFE933A  
        add     #h'7F, r2            #h'7F + r2, store in r2, r2 = h'7F  
        mov.l   #off\_4B71C, r4      #transfer long address of off\_4B71C into r4, r4 points to address FFFE87D8  
        add     #1, r2                #1 + r2, store in r2, r2 = h'80  
        mov.b   @(2,r1), r0            #use r1's pointer + 2 bytes for new address, then transfer address to r0 = FFFE933C  
        mov.w   r2, @(0,r4)          #transfer word r2 into r4's pointer + offset 0, r4 = 80  
        mov.l   off\_4B720, r4        #transfer long address of unk\_105C4, the location of the data array with 21 bytes, r4 = 105C3, pointer to data table, input  
        mov.l   off\_4B724, r10        #transfer long FUNC\_UNK1 address into r10, r10 = FUNC\_UNK2\_00440+36p  
        mov.b   r12, @(0,r8)        #transfer byte from r12 into r8, r8 = ??  
        jsr     @r10                #jump to FUNC\_UNK2\_00440+36p at address specified at r10 after branch delay  
        extu.b  r0, r5                #extend unsigned r0 with zeros from address r5, r0 = 93, r5 = ?  
        mov.b   @(0,r8), r14        #use r8's pointer + 0 and transfer byte into r14  
        extu.b  r0, r0                #extend unsigned r0 with zeros from bit8 to bit 31  
        extu.b  r14, r12            #extend unsigned r14 with zeros from byte r12, r12 =  
        mul.l   r0, r12                #r12 x r0 store in MAC  
  
# ---------------------------------------------------------------------------  
  
off\_4AF68:      .long byte\_FFFE8FE2        #4AF68 = long byte FFFE8FE2

off\_4B718:      .long byte\_FFFE933A        #4B718 = long byte FFFE933A

off\_4B71C:      .long unk\_FFFE87D8        #4B71C = long unk FFFE87D8

off\_4B720:      .long unk\_105C3            #4B720 = long unk 105C3, input  
off\_4B724:      .long FUNC\_UNK1            #4B724 = long FUNC\_UNK1,   
  
   