Device (RTEK)   
{   
Name (\_ADR, Zero) // \_ADR: Address   
Name (\_HID, "ESSX8316") // \_HID: Hardware ID   
Name (\_CID, "ESSX8316") // \_CID: Compatible ID   
Name (\_DDN, "ES8316") // \_DDN: DOS Device Name   
Name (\_UID, One) // \_UID: Unique ID   
  
Method (\_CRS, 0, NotSerialized) // \_CRS: Current Resource Settings   
{   
Name (SBUF, ResourceTemplate ()   
{   
I2cSerialBus (0x0011, ControllerInitiated, 0x00061A80,   
AddressingMode7Bit, "[\\\_SB.I2C2](file:///\\_SB.I2C2)",   
0x00, ResourceConsumer, ,   
)   
GpioIo(Exclusive, PullDefault, 0, 0, IoRestrictionOutputOnly, "[\\\_SB.GPO2](file:///\\_SB.GPO2)",0x00, ResourceConsumer, , )   
{   
0x0015   
}//GPIOS[21]   
GpioInt (Edge, ActiveBoth, Exclusive, PullNone, 0x0000,   
"[\\\_SB.GPO2](file:///\\_SB.GPO2)", 0x00, ResourceConsumer, ,   
)   
{ // Pin list   
0x0004   
}   
})   
Return (SBUF)   
}   
Method (\_STA, 0, NotSerialized) // \_STA: Status   
{   
Return (0x0F)   
}   
Method (\_PS3, 0, NotSerialized) // \_PS3: Power State 3   
{   
Store (Zero, CKC3)   
}   
Method (\_PS0, 0, NotSerialized) // \_PS0: Power State 0   
{   
Store (One, CKC3)   
}   
Method (\_DIS, 0, NotSerialized) // \_DIS: Disable Device   
{   
}   
#define PLATFORM\_MAINMIC\_TYPE 0x00   
#define PLATFORM\_HPMIC\_TYPE 0x01   
#define PLATFORM\_SPK\_TYPE 0x02   
#define PLATFORM\_HPDET\_INV 0x03

#define PLATFORM\_MIC\_DE\_POP 0x06

#define PLATFORM\_SUPPORT\_DSM 0x0F   
  
#define HP\_CODEC\_LINEIN\_PGA\_GAIN 0x10   
#define MAIN\_CODEC\_LINEIN\_PGA\_GAIN 0x20   
  
#define HP\_CODEC\_D2SEPGA\_GAIN 0x11   
#define MAIN\_CODEC\_D2SEPGA\_GAIN 0x21   
  
#define HP\_CODEC\_ADC\_VOLUME 0x12   
#define MAIN\_CODEC\_ADC\_VOLUME 0x22   
  
#define HP\_CODEC\_ADC\_ALC\_ENABLE 0x13   
#define MAIN\_CODEC\_ADC\_ALC\_ENABLE 0x23   
  
#define HP\_CODEC\_ADC\_ALC\_TARGET\_LEVEL 0x14   
#define MAIN\_CODEC\_ADC\_ALC\_TARGET\_LEVEL 0x24   
  
#define HP\_CODEC\_ADC\_ALC\_MAXGAIN 0x15   
#define MAIN\_CODEC\_ADC\_ALC\_MAXGAIN 0x25   
  
#define HP\_CODEC\_ADC\_ALC\_MINGAIN 0x16   
#define MAIN\_CODEC\_ADC\_ALC\_MINGAIN 0x26   
  
#define HP\_CODEC\_ADC\_ALC\_HLDTIME 0x17   
#define MAIN\_CODEC\_ADC\_ALC\_HLDTIME 0x27   
  
#define HP\_CODEC\_ADC\_ALC\_DCYTIME 0x18   
#define MAIN\_CODEC\_ADC\_ALC\_DCYTIME 0x28   
  
#define HP\_CODEC\_ADC\_ALC\_ATKTIME 0x19   
#define MAIN\_CODEC\_ADC\_ALC\_ATKTIME 0x29   
  
#define HP\_CODEC\_ADC\_ALC\_NGTYPE 0x1a   
#define MAIN\_CODEC\_ADC\_ALC\_NGTYPE 0x2a   
  
#define HP\_CODEC\_ADC\_ALC\_NGTHLD 0x1b   
#define MAIN\_CODEC\_ADC\_ALC\_NGTHLD 0x2b

#define MAIN\_CODEC\_ADC\_GUI\_STEP 0x2C

#define MAIN\_CODEC\_ADC\_GUI\_GAIN\_RANGE 0x2D

#define HP\_CODEC\_DAC\_HPMIX\_HIGAIN 0x40   
#define SPK\_CODEC\_DAC\_HPMIX\_HIGAIN 0x50   
  
#define HP\_CODEC\_DAC\_HPMIX\_VOLUME 0x41   
#define SPK\_CODEC\_DAC\_HPMIX\_VOLUME 0x51   
  
#define HP\_CODEC\_DAC\_HPOUT\_VOLUME 0x42   
#define SPK\_CODEC\_DAC\_HPOUT\_VOLUME 0x52   
  
#define HP\_CODEC\_LDAC\_VOLUME 0x44   
#define HP\_CODEC\_RDAC\_VOLUME 0x54   
  
#define SPK\_CODEC\_LDAC\_VOLUME 0x45   
#define SPK\_CODEC\_RDAC\_VOLUME 0x55   
  
#define HP\_CODEC\_DAC\_AUTOMUTE 0x46   
#define SPK\_CODEC\_DAC\_AUTOMUTE 0x56   
  
#define HP\_CODEC\_DAC\_MONO 0x4A   
#define SPK\_CODEC\_DAC\_MONO 0x5A   
  
Method (\_DSM, 4, NotSerialized) // \_DSM: Device-Specific Method   
{   
If (LEqual (Arg0, ToUUID ("a9800c04-e016-343e-41f4-6bcce70f4332")))   
{   
If (LEqual (Arg2, PLATFORM\_MAINMIC\_TYPE)) //MAIN MIC TYPE   
{   
//Return (0xAA) // DMIC AT HIGH LEVEL   
//Return (0x55) // DMIC AT LOW LEVEL   
//Return (0xBB) // AMIC at lin1rin1   
Return (0xcc) // AMIC at LIN2RIN2   
}   
  
If (LEqual (Arg2, PLATFORM\_HPMIC\_TYPE)) //HP MIC TYPE   
{   
//Return (0xAA) // DMIC AT HIGH LEVEL   
//Return (0x55) // DMIC AT LOW LEVEL   
Return (0xBB) // AMIC at lin1rin1   
//return (0xcc) // AMIC at LIN2RIN2   
}   
  
If (LEqual (Arg2, PLATFORM\_SPK\_TYPE)) //SPEAKER TYPE   
{   
Return (0x02) //stereo speaker type   
//Return (0x01) //mono speaker type   
//Return (0x00) //no speaker   
}   
  
If (LEqual (Arg2, PLATFORM\_HPDET\_INV))   
{   
Return (0x00) // Normal   
//Return (0x01) // Inverted   
}

If (LEqual (Arg2, PLATFORM\_MIC\_DE\_POP))   
{   
Return (0x00) // no depop

//Return (0x01) // de pop   
}

If (LEqual (Arg2, PLATFORM\_SUPPORT\_DSM))   
{   
//Return (0x00) // Don't support DSM Data   
return (0x01) // Support DSM Data   
}   
  
If (LEqual (Arg2, HP\_CODEC\_LINEIN\_PGA\_GAIN))   
{   
//Return (0x00) //gain =0db   
//Return (0x01) //gain = +3db   
//Return (0x02) //gain = +6db   
//Return (0x03) //gain = +9db   
//Return (0x04) //gain = +12db   
//Return (0x05) //gain = +15db   
return (0x06) //gain = +18db   
//Return (0x07) //gain = +21db   
//Return (0x08) //gain = +24db   
//Return (0x09) //gain = +27db   
//Return (0x0a) //gain = +30db   
}   
If (LEqual (Arg2, MAIN\_CODEC\_LINEIN\_PGA\_GAIN))   
{   
//Return (0x00) //gain =0db   
//Return (0x01) //gain = +3db   
//Return (0x02) //gain = +6db   
//Return (0x03) //gain = +9db   
//Return (0x04) //gain = +12db   
//Return (0x05) //gain = +15db   
//return (0x06) //gain = +18db   
//Return (0x07) //gain = +21db   
Return (0x08) //gain = +24db   
//Return (0x09) //gain = +27db   
//Return (0x0a) //gain = +30db   
}

If (LEqual (Arg2, MAIN\_CODEC\_ADC\_GUI\_STEP))   
{   
Return (0x03) //3db / step for mic boost gain  
//Return (0x06) //6db /step for mic boost gain

//Return(0x0a) //10db /step for mic boost gain

}

If (LEqual (Arg2, MAIN\_CODEC\_ADC\_GUI\_GAIN\_RANGE))   
{   
//definition for mic boost gain range, maximum gain  
Return(0x1B) // 27DB GAIN RANGE

}

If (LEqual (Arg2, HP\_CODEC\_D2SEPGA\_GAIN))   
{   
Return (0x01) //gain = +15db   
//Return (0x00) //gain = 0db   
}   
If (LEqual (Arg2, MAIN\_CODEC\_D2SEPGA\_GAIN))   
{   
Return (0x01) //gain = +15db   
//Return (0x00) //gain = 0db   
}   
  
If (LEqual (Arg2, HP\_CODEC\_ADC\_VOLUME))   
{   
Return (0x00) // volume = 0db, -0.5db/setp, 0xc0 <-> -96db   
}   
If (LEqual (Arg2, MAIN\_CODEC\_ADC\_VOLUME))   
{   
Return (0x00) // volume = 0db, -0.5db/setp, 0xc0 <-> -96db   
}   
  
If (LEqual (Arg2, HP\_CODEC\_ADC\_ALC\_ENABLE))   
{   
Return (0x01) //enable ALC   
//Return (0x00) //Disable ALC   
}   
If (LEqual (Arg2, MAIN\_CODEC\_ADC\_ALC\_ENABLE))   
{   
Return (0x01) //enable ALC   
//Return (0x00) //Disable ALC   
}   
  
if (LEqual (Arg2, HP\_CODEC\_ADC\_ALC\_TARGET\_LEVEL))   
{   
//Return (0x00) //gain = -16.5db   
//Return (0x01) //gain = -15db   
//Return (0x02) //gain = -13.5db   
//Return (0x03) //gain = -12db   
//Return (0x04) //gain = -10.5db   
//Return (0x05) //gain = -9db   
//return (0x06) //gain = -7.5db   
//Return (0x07) //gain = -6db   
//Return (0x08) //gain = -4.5db   
//Return (0x09) //gain = -3db   
return (0x0a) //gain = -1.5db   
}   
if (LEqual (Arg2, MAIN\_CODEC\_ADC\_ALC\_TARGET\_LEVEL))   
{   
//Return (0x00) //gain = -16.5db   
//Return (0x01) //gain = -15db   
//Return (0x02) //gain = -13.5db   
//Return (0x03) //gain = -12db   
//Return (0x04) //gain = -10.5db   
//Return (0x05) //gain = -9db   
//return (0x06) //gain = -7.5db   
//Return (0x07) //gain = -6db   
//Return (0x08) //gain = -4.5db   
//Return (0x09) //gain = -3db   
return (0x0a) //gain = -1.5db   
}   
  
if (LEqual (Arg2, HP\_CODEC\_ADC\_ALC\_MAXGAIN))   
{   
//Return (0x00) //gain = -6.5db   
//Return (0x01) //gain = -5db   
//Return (0x02) //gain = -3.5db   
//Return (0x03) //gain = -2db   
//Return (0x04) //gain = -0.5db   
//Return (0x05) //gain = +1db   
//return (0x06) //gain = +2.5db   
//Return (0x07) //gain = +4db   
//Return (0x08) //gain = +5.5db   
//Return (0x09) //gain = +7db   
//return (0x0a) //gain = +8.5db   
//Return (0x0b) //gain = +10db   
//Return (0x0c) //gain = +11.5db   
//Return (0x0d) //gain = +13db   
//Return (0x0e) //gain = +14.5db   
//Return (0x0f) //gain = +16db   
//Return (0x10) //gain = +17.5db   
//return (0x11) //gain = +19db   
return (0x12) //gain = +20.5db   
//Return (0x13) //gain = +22db   
//Return (0x14) //gain = +23.5db   
//Return (0x15) //gain = +25db   
//return (0x16) //gain = +26.5db   
//Return (0x17) //gain = +28db   
//Return (0x18) //gain = +29.5db   
//Return (0x19) //gain = +31db   
//Return (0x1a) //gain = +32.5db   
//Return (0x1b) //gain = +34db   
//Return (0x1c) //gain = +35.5db   
}   
if (LEqual (Arg2, MAIN\_CODEC\_ADC\_ALC\_MAXGAIN))   
{   
//Return (0x00) //gain = -6.5db   
//Return (0x01) //gain = -5db   
//Return (0x02) //gain = -3.5db   
//Return (0x03) //gain = -2db   
//Return (0x04) //gain = -0.5db   
//Return (0x05) //gain = +1db   
//return (0x06) //gain = +2.5db   
//Return (0x07) //gain = +4db   
//Return (0x08) //gain = +5.5db   
//Return (0x09) //gain = +7db   
//return (0x0a) //gain = +8.5db   
//Return (0x0b) //gain = +10db   
//Return (0x0c) //gain = +11.5db   
//Return (0x0d) //gain = +13db   
//Return (0x0e) //gain = +14.5db   
//Return (0x0f) //gain = +16db   
//Return (0x10) //gain = +17.5db   
//return (0x11) //gain = +19db   
return (0x12) //gain = +20.5db   
//Return (0x13) //gain = +22db   
//Return (0x14) //gain = +23.5db   
//Return (0x15) //gain = +25db   
//return (0x16) //gain = +26.5db   
//Return (0x17) //gain = +28db   
//Return (0x18) //gain = +29.5db   
//Return (0x19) //gain = +31db   
//Return (0x1a) //gain = +32.5db   
//Return (0x1b) //gain = +34db   
//Return (0x1c) //gain = +35.5db   
}   
  
if (LEqual (Arg2, HP\_CODEC\_ADC\_ALC\_MINGAIN))   
{   
//Return (0x00) //gain = -12db   
//Return (0x01) //gain = -10.5db   
//Return (0x02) //gain = -9db   
//Return (0x03) //gain = -7.5db   
//Return (0x04) //gain = -6db   
//Return (0x05) //gain = -4.51db   
//return (0x06) //gain = -3db   
//Return (0x07) //gain = -1.5db   
return (0x08) //gain = 0db   
//Return (0x09) //gain = +1.5db   
//return (0x0a) //gain = +3db   
//Return (0x0b) //gain = +4.5db   
//Return (0x0c) //gain = +6db   
//Return (0x0d) //gain = +7.5db   
//Return (0x0e) //gain = +9db   
//Return (0x0f) //gain = +10.5db   
//Return (0x10) //gain = +12db   
//return (0x11) //gain = +13.5db   
//return (0x12) //gain = +15db   
//Return (0x13) //gain = +16.5db   
//Return (0x14) //gain = +18db   
//Return (0x15) //gain = +19.5db   
//return (0x16) //gain = +21db   
//Return (0x17) //gain = +22.5db   
//Return (0x18) //gain = +24db   
//Return (0x19) //gain = +25.5db   
//Return (0x1a) //gain = +27db   
//Return (0x1b) //gain = +28.5db   
//Return (0x1c) //gain = +30db   
}   
if (LEqual (Arg2, MAIN\_CODEC\_ADC\_ALC\_MINGAIN))   
{   
//Return (0x00) //gain = -12db   
//Return (0x01) //gain = -10.5db   
//Return (0x02) //gain = -9db   
//Return (0x03) //gain = -7.5db   
//Return (0x04) //gain = -6db   
//Return (0x05) //gain = -4.51db   
//return (0x06) //gain = -3db   
//Return (0x07) //gain = -1.5db   
return (0x08) //gain = 0db   
//Return (0x09) //gain = +1.5db   
//return (0x0a) //gain = +3db   
//Return (0x0b) //gain = +4.5db   
//Return (0x0c) //gain = +6db   
//Return (0x0d) //gain = +7.5db   
//Return (0x0e) //gain = +9db   
//Return (0x0f) //gain = +10.5db   
//Return (0x10) //gain = +12db   
//return (0x11) //gain = +13.5db   
//return (0x12) //gain = +15db   
//Return (0x13) //gain = +16.5db   
//Return (0x14) //gain = +18db   
//Return (0x15) //gain = +19.5db   
//return (0x16) //gain = +21db   
//Return (0x17) //gain = +22.5db   
//Return (0x18) //gain = +24db   
//Return (0x19) //gain = +25.5db   
//Return (0x1a) //gain = +27db   
//Return (0x1b) //gain = +28.5db   
//Return (0x1c) //gain = +30db   
}   
  
if (LEqual (Arg2, HP\_CODEC\_ADC\_ALC\_HLDTIME))   
{   
return (0x00) //time = 0ms   
//Return (0x01) //time = 2.67ms   
//Return (0x02) //time = 5.33ms   
//Return (0x03) //time = 10.66ms   
//Return (0x04) //time = 21.32ms   
//Return (0x05) //time = 42.64ms   
//return (0x06) //time = 85.38ms   
//Return (0x07) //time = 170.76ms   
//Return (0x08) //time = 341.52ms   
//Return (0x09) //time = 0.68s   
//return (0x0a) //time = 1.36s   
}   
if (LEqual (Arg2, MAIN\_CODEC\_ADC\_ALC\_HLDTIME))   
{   
return (0x00) //time = 0ms   
//Return (0x01) //time = 2.67ms   
//Return (0x02) //time = 5.33ms   
//Return (0x03) //time = 10.66ms   
//Return (0x04) //time = 21.32ms   
//Return (0x05) //time = 42.64ms   
//return (0x06) //time = 85.38ms   
//Return (0x07) //time = 170.76ms   
//Return (0x08) //time = 341.52ms   
//Return (0x09) //time = 0.68s   
//return (0x0a) //time = 1.36s   
}   
  
if (LEqual (Arg2, HP\_CODEC\_ADC\_ALC\_DCYTIME))   
{   
return (0x00) //time = 410us   
//Return (0x01) //time = 820us   
//Return (0x02) //time = 1.64ms   
//Return (0x03) //time = 3.28ms   
//Return (0x04) //time = 6.56ms   
//Return (0x05) //time = 13.12ms   
//return (0x06) //time = 26.24ms   
//Return (0x07) //time = 52.48ms   
//Return (0x08) //time = 104.96ms   
//Return (0x09) //time = 209.92ms   
//return (0x0a) //time = 420ms   
}   
if (LEqual (Arg2, MAIN\_CODEC\_ADC\_ALC\_DCYTIME))   
{   
return (0x00) //time = 410us   
//Return (0x01) //time = 820us   
//Return (0x02) //time = 1.64ms   
//Return (0x03) //time = 3.28ms   
//Return (0x04) //time = 6.56ms   
//Return (0x05) //time = 13.12ms   
//return (0x06) //time = 26.24ms   
//Return (0x07) //time = 52.48ms   
//Return (0x08) //time = 104.96ms   
//Return (0x09) //time = 209.92ms   
//return (0x0a) //time = 420ms   
}   
  
if (LEqual (Arg2, HP\_CODEC\_ADC\_ALC\_ATKTIME))   
{   
//return (0x00) //time = 104us   
//Return (0x01) //time = 208us   
return (0x02) //time = 416ms   
//Return (0x03) //time = 832ms   
//Return (0x04) //time = 1.664ms   
//Return (0x05) //time = 3.328ms   
//return (0x06) //time = 6.656ms   
//Return (0x07) //time = 13.312ms   
//Return (0x08) //time = 26.624ms   
//Return (0x09) //time = 53.248ms   
//return (0x0a) //time = 106.496ms   
}   
if (LEqual (Arg2, MAIN\_CODEC\_ADC\_ALC\_ATKTIME))   
{   
//return (0x00) //time = 104us   
//Return (0x01) //time = 208us   
return (0x02) //time = 416ms   
//Return (0x03) //time = 832ms   
//Return (0x04) //time = 1.664ms   
//Return (0x05) //time = 3.328ms   
//return (0x06) //time = 6.656ms   
//Return (0x07) //time = 13.312ms   
//Return (0x08) //time = 26.624ms   
//Return (0x09) //time = 53.248ms   
//return (0x0a) //time = 106.496ms   
}   
  
if (LEqual (Arg2, HP\_CODEC\_ADC\_ALC\_NGTYPE))   
{   
//return (0x00) //noise gate disable   
//Return (0x01) //noise gate enable, hold gain type   
return (0x03) //noise gate enable, mute type   
}   
if (LEqual (Arg2, MAIN\_CODEC\_ADC\_ALC\_NGTYPE))   
{   
//return (0x00) //noise gate disable   
//Return (0x01) //noise gate enable, hold gain type   
return (0x03) //noise gate enable, mute type   
}   
  
if (LEqual (Arg2, HP\_CODEC\_ADC\_ALC\_NGTHLD))   
{   
//Return (0x00) //Thereshold = -76.5db   
return (0x01) //Thereshold = -75db   
//Return (0x02) //Thereshold = -73.5db   
//Return (0x03) //Thereshold = -72db   
//Return (0x04) //Thereshold = -70.5db   
//Return (0x05) //Thereshold = -69db   
//return (0x06) //Thereshold = -67.5db   
//Return (0x07) //Thereshold = -66db   
//return (0x08) //Thereshold = -64.5db   
//Return (0x09) //Thereshold = -63db   
//return (0x0a) //Thereshold = -61.5db   
//Return (0x0b) //Thereshold = -60db   
//Return (0x0c) //Thereshold = -58.5db   
//Return (0x0d) //Thereshold = -57db   
//Return (0x0e) //Thereshold = -55.5db   
//Return (0x0f) //Thereshold = -54db   
//Return (0x10) //Thereshold = -52.5db   
//return (0x11) //Thereshold = -51db   
//return (0x12) //Thereshold = -49.5db   
//Return (0x13) //Thereshold = -48db   
//Return (0x14) //Thereshold = -46.5db   
//Return (0x15) //Thereshold = -45db   
//return (0x16) //Thereshold = -43.5db   
//Return (0x17) //Thereshold = -42db   
//Return (0x18) //Thereshold = -40.5db   
//Return (0x19) //Thereshold = -39db   
//Return (0x1a) //Thereshold = -37.5db   
//Return (0x1b) //Thereshold = -36db   
//Return (0x1c) //Thereshold = -34.5db   
//Return (0x1d) //Thereshold = -33db   
//Return (0x1e) //Thereshold = -31.5db   
//Return (0x1f) //Thereshold = -30db   
}   
if (LEqual (Arg2, MAIN\_CODEC\_ADC\_ALC\_NGTHLD))   
{   
//Return (0x00) //Thereshold = -76.5db   
return (0x01) //Thereshold = -75db   
//Return (0x02) //Thereshold = -73.5db   
//Return (0x03) //Thereshold = -72db   
//Return (0x04) //Thereshold = -70.5db   
//Return (0x05) //Thereshold = -69db   
//return (0x06) //Thereshold = -67.5db   
//Return (0x07) //Thereshold = -66db   
//return (0x08) //Thereshold = -64.5db   
//Return (0x09) //Thereshold = -63db   
//return (0x0a) //Thereshold = -61.5db   
//Return (0x0b) //Thereshold = -60db   
//Return (0x0c) //Thereshold = -58.5db   
//Return (0x0d) //Thereshold = -57db   
//Return (0x0e) //Thereshold = -55.5db   
//Return (0x0f) //Thereshold = -54db   
//Return (0x10) //Thereshold = -52.5db   
//return (0x11) //Thereshold = -51db   
//return (0x12) //Thereshold = -49.5db   
//Return (0x13) //Thereshold = -48db   
//Return (0x14) //Thereshold = -46.5db   
//Return (0x15) //Thereshold = -45db   
//return (0x16) //Thereshold = -43.5db   
//Return (0x17) //Thereshold = -42db   
//Return (0x18) //Thereshold = -40.5db   
//Return (0x19) //Thereshold = -39db   
//Return (0x1a) //Thereshold = -37.5db   
//Return (0x1b) //Thereshold = -36db   
//Return (0x1c) //Thereshold = -34.5db   
//Return (0x1d) //Thereshold = -33db   
//Return (0x1e) //Thereshold = -31.5db   
//Return (0x1f) //Thereshold = -30db   
}   
  
if (LEqual (Arg2, HP\_CODEC\_DAC\_HPMIX\_HIGAIN))   
{   
return (0x00) //gain = 0db   
//return (0x88) //gain = -6db   
}   
if (LEqual (Arg2, SPK\_CODEC\_DAC\_HPMIX\_HIGAIN))   
{   
return (0x00) //gain = 0db   
//return (0x88) //gain = -6db   
}   
  
if (LEqual (Arg2, HP\_CODEC\_DAC\_HPMIX\_VOLUME))   
{   
//Return (0x00) //VOLUME = -12db   
//Return (0x11) //VOLUME = -10.5db   
//Return (0x22) //VOLUME = -9db   
//return (0x33) //VOLUME = -7.5db   
//Return (0x44) //VOLUME = -6db   
//return (0x88) //VOLUME = -4.5db   
//Return (0x99) //VOLUME = -3db   
//return (0xaa) //VOLUME = -1.5db   
return (0xbb) //Thereshold = 0db   
}   
if (LEqual (Arg2, SPK\_CODEC\_DAC\_HPMIX\_VOLUME))   
{   
//Return (0x00) //VOLUME = -12db   
//Return (0x11) //VOLUME = -10.5db   
//Return (0x22) //VOLUME = -9db   
//return (0x33) //VOLUME = -7.5db   
//Return (0x44) //VOLUME = -6db   
//return (0x88) //VOLUME = -4.5db   
//Return (0x99) //VOLUME = -3db   
//return (0xaa) //VOLUME = -1.5db   
return (0xbb) //Thereshold = 0db   
}   
  
if (LEqual (Arg2, HP\_CODEC\_DAC\_HPOUT\_VOLUME))   
{   
return (0x00) //VOLUME = -0db   
//Return (0x11) //VOLUME = -12db   
//Return (0x22) //VOLUME = -24db   
//return (0x33) //VOLUME = -48db   
}   
if (LEqual (Arg2, SPK\_CODEC\_DAC\_HPOUT\_VOLUME))   
{   
return (0x00) //VOLUME = -0db   
//Return (0x11) //VOLUME = -12db   
//Return (0x22) //VOLUME = -24db   
//return (0x33) //VOLUME = -48db   
}   
  
if (LEqual (Arg2, HP\_CODEC\_LDAC\_VOLUME))   
{   
Return (0x00) // volume = 0db, -0.5db/setp, 0xc0 <-> -96db   
}   
if (LEqual (Arg2, HP\_CODEC\_RDAC\_VOLUME))   
{   
Return (0x00) // volume = 0db, -0.5db/setp, 0xc0 <-> -96db   
}   
if (LEqual (Arg2, SPK\_CODEC\_LDAC\_VOLUME))   
{   
Return (0x00) // volume = 0db, -0.5db/setp, 0xc0 <-> -96db   
}   
if (LEqual (Arg2, SPK\_CODEC\_RDAC\_VOLUME))   
{   
Return (0x00) // volume = 0db, -0.5db/setp, 0xc0 <-> -96db   
}   
  
if (LEqual (Arg2, HP\_CODEC\_DAC\_MONO))   
{   
Return (0x00) // stereo, please refer to register0x32.bit3   
//Return (0x01) //mono, (L+R)/2   
}   
if (LEqual (Arg2, SPK\_CODEC\_DAC\_MONO))   
{   
Return (0x00) // stereo, please refer to register0x32.bit3   
//Return (0x01) //mono, (L+R)/2   
}   
  
if (LEqual (Arg2, HP\_CODEC\_DAC\_AUTOMUTE))   
{   
Return (0x00) // no automute, please refer to register0x31.bit5 and reg0x50.bit3   
//Return (0x01) //digital mute   
//Return (0x02) //analog mute   
}   
if (LEqual (Arg2, SPK\_CODEC\_DAC\_AUTOMUTE))   
{   
//Return (0x00) // no automute, please refer to register0x31.bit5 and reg0x50.bit3   
Return (0x01) //digital mute   
//Return (0x02) //analog mute   
}   
}   
  
Return (0xFF)   
}   
}