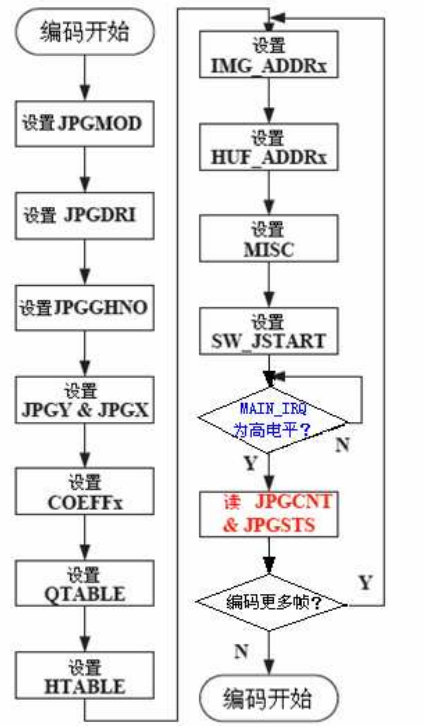

图 22-4 基本编码的流程图实例

基本的 JPEG 编码采取下列步骤：

（1）在 JPGMOD 寄存器中，设置流程模式为编码流程和亚抽样模式。

（2）设置 MCU 和 RST 标记寄存器 JPGDRI。

（3）设置 Q 和 H 表数目寄存器 JPGQHNO。

（4）设置 JPGY 和 JPGX 寄存器。

（5）设置系数寄存器 COEFF1, COEFF2, COEFF3，用于色彩空间转换。

（6）设置 QTABLE 和 HTABLE。

（7）设置源图像数据，第一帧地址寄存器 IMG_ADDR0 和第二帧的 IMG_ADDR1。

（8）设置 JPEG 目的文件地址寄存器 HUFADDR0 和下一编码 JPEG 文件地址寄存器 HUFADDR1。

（9）设置混合寄存器 MISC（设置 EMS 位为 0，选择 MODE_SEL 0x1 或 0x2）。

（10）设置 SW_JSTART 为高电平。

（11）必须读 JPGIRQ 和 JPGSTS 寄存器，以清除内部等待中断。

代码实现：

具体在 ARM11 中 JPEG 基本的编码，代码编写如下：

```
void JPEG_InitRegsForEncoding(
    u32 uRawHsz, u32 uRawVsz, u32 uRawAddr, CSPACE eRawType,
    u32 uJpgAddr, JPEG_TYPE uJpgType, bool bIsOnTheFly, bool bIsMotion)
{
    int i;
    Assert(eRawType == YCBYCR || eRawType == RGB16);
    Outp32(JPGMOD, (uJpgType == JPEG_422)? (0x1<<0) : (0x2<<0)); // Encoded to yuv422 or yuv420
    Outp32(JPGDRI, 2); // MCU inserts RST marker
    Outp32(JPGQHNO, 0x0);
    Outp32(JPGX, uRawHsz);
    Outp32(JPGY, uRawVsz);
    Outp32(JIMGADDR0, uRawAddr); // Address of input image
    Outp32(JHUFADDR0, uJpgAddr); // Address of JPEG stream
    Outp32(JIMGADDR1, uRawAddr); // Address of input image
    Outp32(JHUFADDR1, uJpgAddr); // next address of motion JPEG stream
    Outp32(JCOEF1, COEF1_RGB_2_YUV); // Coefficient value 1 for RGB to YCbCr
    Outp32(JCOEF2, COEF2_RGB_2_YUV); // Coefficient value 2 for RGB to YCbCr
    Outp32(JCOEF3, COEF3_RGB_2_YUV); // Coefficient value 3 for RGB to YCbCr
    Outp32(JMISC,
        (bIsOnTheFly ? 0 : (eRawType == YCBYCR ? 1 : 2))<<5 |
        (bIsOnTheFly ? 1 : 0)<<2
        );
    Outp32(JPG_CON,      (bIsMotion ? ENABLE_MOTION_ENC : DISABLE_MOTION_ENC));

//  Outp32(JPGIRQEN, 0x10);
//  Outp32(JPGIRQEN, 1<<4); // Deleted @2006.6.8
    // Quantiazation and Huffman Table setting
    //---------------------------------------
```

```
    for (i=0; i<64; i++)
        Outp32((JQTBL0+i*4), (u32)QTBL0[i]);
    for (i=0; i<64; i++)
        Outp32((JQTBL1+i*4), (u32)std_chrominance_quant_tbl_plus[i]);
    for (i=0; i<16; i++)
        Outp32((JHDCTBL0+i*4), (u32)HDCTBL0[i]);
    for (i=0; i<12; i++)
        Outp32((JHDCTBLG0+i*4), (u32)HDCTBLG0[i]);
    for (i=0; i<16; i++)
        Outp32((JHACTBL0+i*4), (u32)HACTBL0[i]);
    for (i=0; i<162; i++)
        Outp32((JHACTBLG0+i*4), (u32)HACTBLG0[i]);
}
```

JPEG 解码顺序：软件控制解码。如图 22-5 所示。

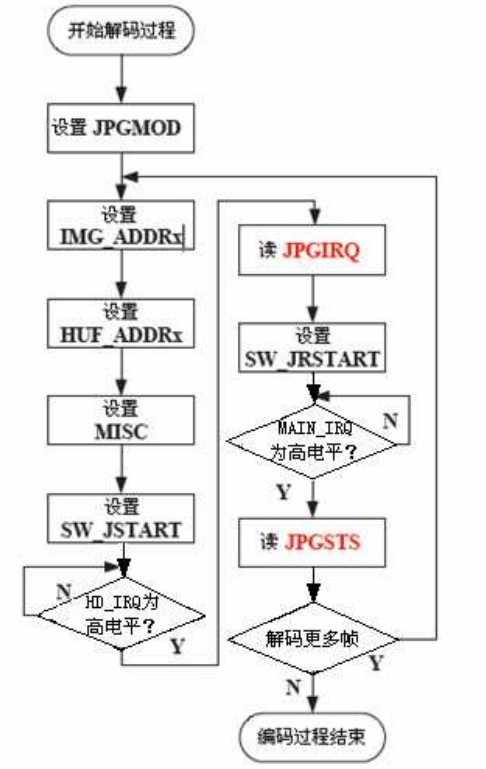

图 22-5 软件控制解码的流程图

采取下列步骤，为软件控制的 JPEG 解码：

（1）在 JPGMOD 寄存器中，设置流程模式为解码流程。

（2）设置第一帧解码图像数据 IMG_ADDR0 和第二帧解码图像数据地址 IMG_ADDR1 的目的地址。

（3）设置第一帧 JPEG 文件 HUFADDR0 和下一 JPEG 文件地址 HUFADDR1 的源地址。

（4）设置混合寄存器 MISC（设置 MODE_SEL 为 0x1 或 0x2 ，DMS 位为 0）。

（5）设置 SW_JSTART 为高电平。

（6）如果 HD_IRQ 为高电平和 ERR_IRQ 为低电平，读 JPGIRQ 寄存器以清除内部等待 IRQ。

（7）设置 SW_JRSTART 为高电平。

（8）如果 MAIN_IRQ 为高电平和 ERR_IRQ 为低电平，从 JPGCNT 寄存器读取帧大小（字节）。

（9）必须读 JPGIRQ 和 JPGSTS 寄存器以清除内部等待中断。

代码实现：

具体在 ARM11 中 JPEG 的解码，代码编写如下：

```
void JPEG_InitRegsForDecoding(
    u32 uSrcAddr, u32 uDstAddr,
    JPEG_DEC_MODE eMode, bool bIncremental, bool bIsMotion
    )
{
    u32 uJpgConVal = 0;
    u32 uMisc = 0;
    if (eMode == HEADER || eMode == HEADER_N_BODY)
    {
        Outp32(JPGMOD, 0x8); // Process mode: Decoding
        Outp32(JHUFADDR0, uSrcAddr); // Address of compresed input data
        Outp32(JHUFADDR1, uSrcAddr); // Address of compresed input data
        Outp32(JPGIRQEN, 0xf<<3); // JPGIRQEN[6:3]=For several error conditions @2006.6.8

        //Outp32(JPGIRQEN, 0xf); // JPGIRQEN[6:3]=For several error conditions @2006.6.8
```

```
        if (eMode == HEADER_N_BODY)
        {
            Outp32(JIMGADDR0, uDstAddr); // Address of decompresed image
            Outp32(JIMGADDR1, uDstAddr); // Address of decompresed image
        }
        uJpgConVal = (eMode == HEADER) ? DISABLE_HW_DEC : ENABLE_HW_DEC;
    }
    else // eMode == BODY
    {
        Outp32(JIMGADDR0, uDstAddr); // Address of decompresed image
        Outp32(JIMGADDR1, uDstAddr); // Address of decompresed image
    }
    if (eMode == BODY || eMode == HEADER_N_BODY)
    {
        uJpgConVal |= (bIsMotion == true) ? ENABLE_MOTION_DEC : DISABLE_MOTION_DEC;
        uMisc = (bIncremental == true) ? INCREMENTAL_DEC : NORMAL_DEC;
    }
    Outp32(JPG_CON, uJpgConVal);
    Outp32(JMISC, uMisc);
}
```

JPEG 解码顺序：硬件控制解码。（只有 1 帧）如图 22-6 所示。

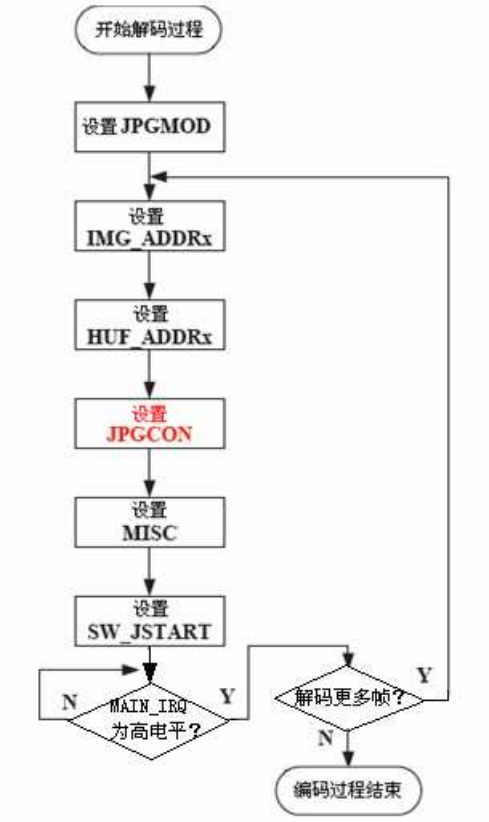

图 22-6 硬件控制解码的流程图

采取下列步骤，硬件控制的 JPEG 解码：

（1）在JPGMOD寄存器中，设置流程模式为解码流程。

（2）设置第一帧解码图像数据IMG_ADDR0和第二帧解码图像数据地址IMG_ADDR1的目的地址。

（3）设置第一帧 JPEG 文件 HUFADDR0 和下一 JPEG 文件地址 HUFADDR1 的源地址。

（4）在 JPGCON 寄存器中，设置 HW_DEC 为 1

MJ_ENC，HW_DEC，CLK_SEL，MJ_DEC

0 1 0 0

（5）设置混合寄存器 MISC（设置 MODE_SEL 为 0x1 或 0x2 ，DMS 位为 0）。

（6）设置 SW_JSTART 为高电平。

（7）如果 MAIN _IRQ 为高电平和 ERR_IRQ 为低电平，从 JPGCNT 寄存器读取帧大小（字节）。

（8）必须读 JPGIRQ 和 JPGSTS 寄存器以清除内部等待中断。

代码实现：

具体在 ARM11 中 JPEG 的解码，代码编写如下：

```
void JPEG_InitRegsForDecoding1(
    u32 uSrcAddr, u32 uDstAddr,
    bool bIsHeaderOnly, bool bIncremental, bool bIsMotion
    )
{
    u32 uJpgConVal = 0;
    Outp32(JPGMOD, 0x8); // Process mode: Decoding
    Outp32(JHUFADDR0, uSrcAddr); // Address of compresed input data
    Outp32(JIMGADDR0, uDstAddr); // Address of decompresed image
    Outp32(JHUFADDR1, uSrcAddr); // Address of compresed input data
    Outp32(JIMGADDR1, uDstAddr); // Address of decompresed image
    Outp32(JPGIRQEN, 0xf<<3); // JPGIRQEN[6:3]=For several error conditions @2006.6.8
#if 0
    Outp32(JPG_CON, (bIsMotion ? ENABLE_MOTION_DEC : DISABLE_MOTION_DEC));
    u32 uJpgConVal;
    jpgInp32(JPG_CON, uJpgConVal);
    if (bIsHeaderOnly)
        uJpgConVal &= ~(1<<2);
    else
        uJpgConVal |= (1<<2);
    Outp32(JPG_CON, uJpgConVal);
#else
    uJpgConVal = (bIsMotion ? ENABLE_MOTION_DEC : DISABLE_MOTION_DEC);
    uJpgConVal |= (bIsHeaderOnly ? DISABLE_HW_DEC : ENABLE_HW_DEC);
    Outp32(JPG_CON, uJpgConVal);
#endif
    if (bIncremental)
        Outp32(JMISC, (1<<3));
```

```
    else
        Outp32(JMISC, (0<<3));
}
```

Motion JPEG 编码顺序，如图 22-7 所示。

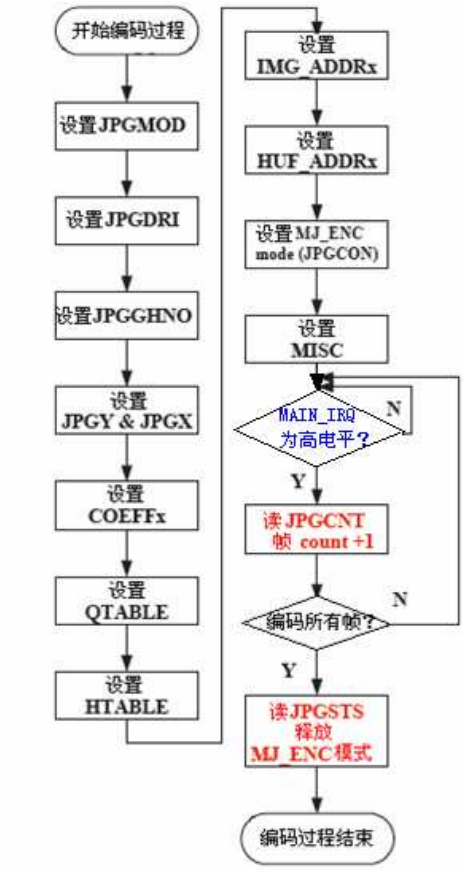

图 22-7　Motion JPEG 编码的流程图

采取下列步骤，为 Motion JPEG 编码：

（1）在JPGMOD寄存器中，设置流程模式为编码流程和亚抽样模式。

（2）设置 MCU 和 RST 标记寄存器 JPGDRI。

（3）设置 Q 和 H 表数目寄存器 JPGQHNO。

（4）设置 JPGY 和 JPGX 寄存器。

（5）设置系数寄存器 COEFF1、COEFF2、COEFF3，用于色彩空间转换。

（6）设置 QTABLE 和 HTABLE。

（7）设置源图像数据，第一帧地址寄存器 IMG_ADDR0 和第二帧的 IMG_ADDR1。

（8）设置 JPEG 目的文件地址寄存器 HUFADDR0 和下一编码 JPEG 文件地址寄存器 HUFADDR1。

（9）在JPGCON 寄存器中，设置运动JPEG编码模式。

MJ_ENC，HW_DEC，CLK_SEL，MJ_DEC

1 0 0 0

（10）设置混合寄存器 MISC（设置 MODE_SEL 为 0x1 或 0x2 ，EMS 位为 0）。

（11）如果 MAIN _IRQ 为高电平和 ERR_IRQ 为低电平，从 JPGCNT 寄存器和增加编码帧数量中，读帧大小（字节）。

（12）如果所有帧都被编码了，读JPGIRQ和JPGSTS 寄存器，以清除内部等待中断。

（13）在 JPGCON 寄存器中，MJ_ENC 被禁用。

代码实现：

具体在 ARM11 中 Motion JPEG 的编码，代码编写如下：

```
    //Motion JPEG 编码
void JPEG_StartEncodingMotionJPEG(
    u16 usHSz, u16 usVSz, u32 uSrcAddr, CSPACE eRawType,
    u32 uDestAddr, JPEG_TYPE eJpgType)
{
    Assert(eRawType == YCBYCR || eRawType == RGB16);

    JPEG_Reset();
    //基本的编码函数 JPEG_InitRegsForEncoding
    JPEG_InitRegsForEncoding(usHSz, usVSz, uSrcAddr, eRawType, uDestAddr, eJpgType, false,
true);
//  DisableMotionEncoding();
    Outp32(JSTART, 0); // 开始
}

void JPEG_InitIpForMotionEncoding(
    u16 uRawHsz, u16 uRawVsz, u32 uRawAddr, CSPACE eRawType,
```

```
    u32 uJpgAddr, JPEG_TYPE eJpgType, u32 uMJpegMaxSize)
{
    Assert(eRawType == YCBYCR || eRawType == RGB16);
    printf(" Enc: x=%d, y=%d, yuv=0x%08x~0x%08x, YUV%d\n",
        uRawHsz, uRawVsz, uRawAddr, uRawAddr+(uRawHsz*uRawVsz*2), (eJpgType == JPEG_422) ? 422:
420);
    JPEG_InitRegsForEncoding(uRawHsz, uRawVsz, uRawAddr, eRawType, uJpgAddr, eJpgType, false,
true); // 配置寄存器
    JPEG_SetNextFrameStartAddr(uJpgAddr+uMJpegMaxSize);
    Outp32(JPGCNT, 0);
    Outp32(JPG_CON, (1<<3)); // 开始 Motion Jpeg 编码
    //+daedoo
    Outp32(JSTART, 0); //开始
}
```

Motion JPEG 解码顺序，如图 22-8 所示。