SIEMENS

Microcontrollers

ApNote AP1617

☐ additional file
APXXXX01.EXE available

Pre- and post-scale 32-Bit-Values for the MDU (Shifting/Rotating of Double-Words)

By using the MDU it is very often necessary to scale the value before or after the calculation to avoid truncation errors during the calculation.

K.H. Mattheis / Siemens HL MCB PD

Shifting / Rotating of Double-Words (32 bit values)

By using the MDU it is very often necessary to scale the value before or after the calculation to avoid truncation errors during the calculation. Scaling is most easily realized by multipling with a factor of 2X which is a simple left shift, e.g. VALUE * 23 >> shift VALUE three(3) positions to the left

The following routines will show you a very effective way for shifting a 32-bit value with the 16-bit ALU using the General Purpose Registers (GPR) more than one bit position. The first routine shows a shift left; the principle used here can also be applied to perform right shifts as well as rotate left or right operations. The last routine shows the rotating of a double-word by 0 to 31 bit positions to the right.

```
Shift_Double_Word:    ; double-word in R2|R1, shift-count in R3

 MOV   R4,R1          ; store(save) low-word in R4
 SHL   R1,R3          ; shift left low-word (R1)
 SHL   R2,R3          ; shift left high-word (R2)
 NEG   R3             ; two's complement of shift-count (R3)
 SHR   R4,R3          ; adjust overflown bits from low-word (R4) to the right
 OR    R2,R4          ; and store to the high-word (R2)
```

Shift a double-word by less than 16 bit positions to the left

```
example:
                      ; double-word:  R2|R1
                      ; 0000 0010 1011 1001   0110 1000 0100 0101
                      ; shift-count: R3 = 6
 MOV   R4,R1          ; R4 = 0110 1000 0100 0101
 SHL   R1,R3          ; R1 = 0001 0001 0100 0000 (0110 10)
 SHL   R2,R3          ; R2 = 1010 1110 0100 0000
 NEG   R3             ; R3 = FFFA (= 10 or -6); only the least 4 significant
                      ; bits are used
 SHR   R4,R3          ; R4 = 0000 0000 0001 1010
 OR    R2,R4          ; R2 = 1010 1110 0101 1010
```

Example for the shift left. The bits shifted from one word to the other are underlined

```
SHL_32:               ; double-word in R2|R1, shift-count in R3

 JNB   R3.4,SHL_32_1 ; check for shift-count ≥ 16
 MOV   R2,R1          ; low-word (R1) in high-word (R2) (shift by 16)
 MOV   R1,#0          ; clear low-word (R1)
 SHL   R2,R3          ; shift left by shift-count - 16 (only the least 4
                      ; bits of the shift count in R3 are evaluated)
 JMPR  cc_UC, SHL_32_Done ;

SHL_32_1:

 MOV   R4,R1        ; store (save) low-word in R4
 SHL   R1,R3        ; shift left low-word (R1)
 SHL   R2,R3        ; shift left high-word (R2)
 NEG   R3           ; two's complement of shift-count (R3)
 SHR   R4,R3        ; adjust overflown bits from low-word to the right
 OR    R2,R4        ; and store to the high-word (R2)

SHL_32_Done:
```

```
...
```

Shift a double-word by 0 to 31 bit positions to the left

```
ROR_32:             ; double-word in R2|R1, rotate-count in R3

 JNB   R3.4,ROR_32_1 ; check for rotate-count ≥ 16
 MOV   R4,R1        ; rotate-count is ≥ 16: exchange words
 MOV   R1,R2        ;
 MOV   R2,R4        ;

ROR_32_1:

 MOV   R4,R1        ; save words
 MOV   R5,R2        ;
 SHR   R1,R3        ; shift the single words by rotate-count - 16 (only
 SHR   R2,R3        ; the least 4 bits of the rotate-count are evaluated)
 NEG   R3           ; two's complement of rotate-count
 SHL   R4,R3        ; shift left saved words by complement of
 SHL   R5,R3        ; rotate-count
 OR    R1,R5        ; combine the shifted parts of the words
 OR    R2,R4        ;
```

Rotate a double-word by 0 to 31 bit positions to the right

```
example:
                    ; double-word:  R2 | R1
                    ; 0000 0010 1011 1001   0110 1000 0100 0101
                    ; rotate-count: R3 = 6
 MOV   R4,R1        ; R4 = 0110 1000 0100 0101
 MOV   R5,R2        ; R5 = 0000 0010 1011 1001
 SHR   R1,R3        ; R1 = 0000 0001 1010 0001
 SHR   R2,R3        ; R2 = 0000 0000 0000 1010
 NEG   R3           ; R3 = FFFA (= -6)
 SHL   R4,R3        ; R4 = 0001 0100 0000 0000
 SHL   R5,R3        ; R5 = 1110 0100 0000 0000
 OR    R1,R5        ; R1 = 1110 0101 1010 0001
 OR    R2,R4        ; R2 = 0001 0100 0000 1010
```

Example for a rotate of < 16 bit positions to the right. The bits rotated from one word into the other are underlined