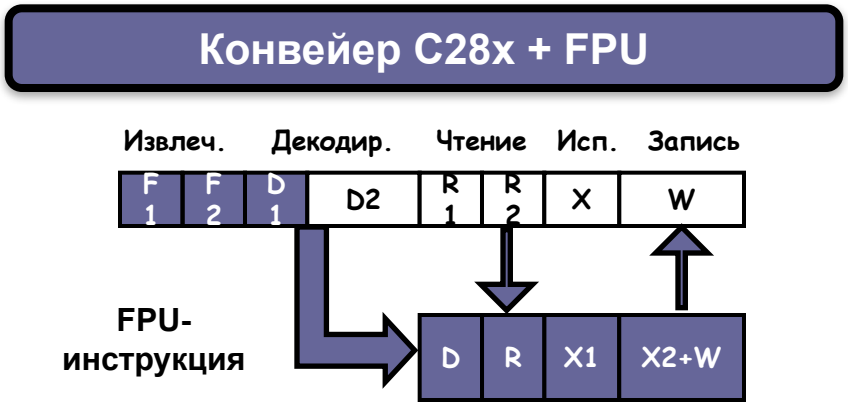
Модуль центрального процессора TMS320F28x

Особенности модуля центрального процессора ЦСП серий Delfino C2833x и C2834x



Серии Delfino C2833x и C2834x содержат модуль FPU (модуль расширения ядра для исполнения инструкций с плавающей точкой – FPU, Floating Point Unit).

- реализована полная поддержка плавающей и фиксированной точки;

- производительность – до 300 MFLOPS на частоте 150 МГц;

- команды с плавающей точкой (FPU-инструкции) используют первую часть конвейера ядра C28x (с фиксированной точкой), а далее исполняются в собственном конвейере;

- команды вычислений с фиксированной точкой направляются на верхний конвейер, а FPU-инструкции – на нижний.

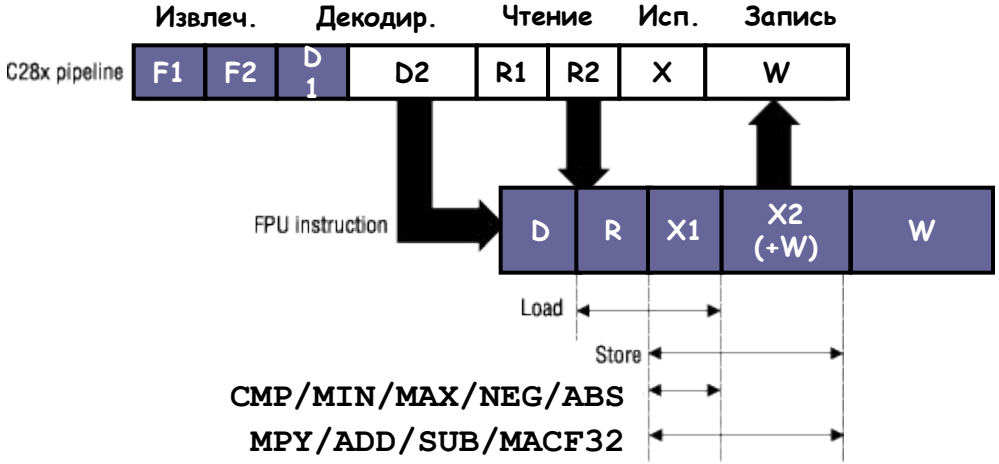
- плавающая точка более надежна в алгоритмах, т.к. исключаются затраты времени на масштабирование данных и устранение избыточности;

- применение плавающей точки позволяет сократить требуемое число тактов на исполнение математических операций до 52%.

Модуль центрального процессора TMS320F28x

Часть команд с плавающей точкой, таких как команды сравнения `CMP`, поиска минимума `MIN`, максимума `MAX`, инвертирования знака числа `NEG` и получения абсолютного значения числа `ABS` являются одноцикловыми.

Другая часть команд, в первую очередь арифметических, – умножения `MPY`, сложения `ADD`, вычитания `SUB`, умножения с накоплением `MACF32` – двуцикловые. Для того, чтобы результат такой операции стал доступен следующей команде, необходимо задержать конвейер на один цикл.



Компилятор C/C++ автоматически оптимизирует алгоритм пользователя с учетом возможного параллельного выполнения команд, добавляя, как правило, «не конфликтующие с конвейером» команды загрузки операндов, которые понадобятся на следующих этапах вычислений. Тем самым практически полностью исключаются непроизводительные задержки конвейера.

Модуль центрального процессора TMS320F28x

```
__sqrt:
    CSB    ACC
    LSLL   ACC,T
    MOVL   XAR6,@ACC
    ASR    AH,#6
    MOVB   @AH,#0xFE,LEQ
    SUB    @AH,#254
    MOVZ   AR0,@AH
    TBIT   @T,#0
    MOV    AH,@T
    LSR    AH,#1
    MOVL   *SP++,ACC
    MOVL   XAR7,#_IQsqrtTable
    MOVL   XAR4,*+XAR7[AR0]
    MOVL   XAR7,#_IQsqrtRoundSatTable
    MOVL   XAR5,*XAR7++
    .if (GLOBAL_Q & 0x0001)==0;
    MOVB   @AR0,#12,NTC
    MOVB   @AR0,#10,TC
    .endif
    .if (GLOBAL_Q & 0x0001)==1
    MOVB   @AR0,#12,TC
    MOVB   @AR0,#8,NTC
    .endif
    MOVL   XT,@XAR4
    QMPYL ACC,XT,@XT
    MOVL   XT,@XAR6
    LSL    ACC,#2
    QMPYL ACC,XT,@ACC
    LSL    ACC,#2
    MOVL   XAR4,@ACC
    MOVL   XT,@XAR4
    QMPYL ACC,XT,@XT
    MOVL   XT,@XAR6
    LSL    ACC,#2
    QMPYL ACC,XT,@ACC
    MOVL   XT,@XAR5
    SUBL   @XT,ACC
    QMPYL ACC,XT,@XAR4
    LSL    ACC,#2
    MOVL   XT,@XAR6
    QMPYL ACC,XT,@ACC
    MOVL   XT,*+XAR7[AR0]
    IMPYL P,XT,@ACC
    QMPYL ACC,XT,@ACC
    .if    GLOBAL_Q >= 24
    LSL64 ACC:P,#((GLOBAL_Q -
    .endif
    .if    GLOBAL_Q <= 21
    ASR64 ACC:P,#((23 - GLOBAL_
    .endif
    MOVL   XT,*--SP
    ASR64 ACC:P,T
    ADD    @PH,#-32768
    ADDCL ACC,*+XAR7[2]
    LRETR
```

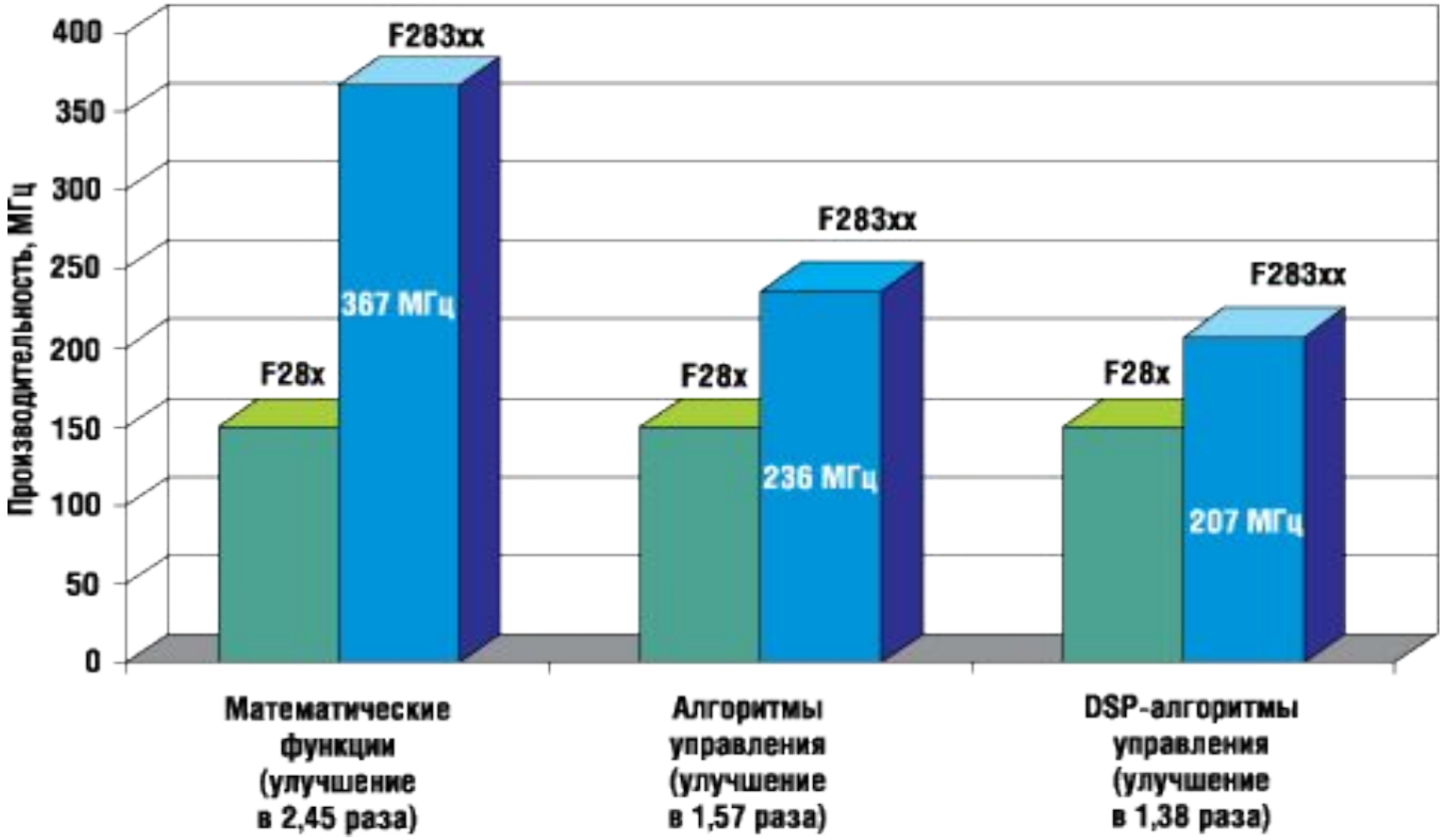
Фикс. тчк.
66 слов
70 тактов

```
__sqrt:
    MOV32    *SP++,R4H
    CMPF32   R0H,#0.0
    MOVST0   ZF,NF
    B   L1,EQ
    EISQRTF32 R1H,R0H
    MOVIZF32  R2H,#0.5
    MPYF32    R2H,R0H,R2H
    MOVIZF32  R3H,#1.5
    MPYF32    R4H,R1H,R2H
    NOP
    MPYF32   R4H,R1H,R4H
    NOP
    SUBF32   R4H,R3H,R4H
    NOP
    MPYF32   R1H,R1H,R4H
    NOP
    MPYF32   R4H,R1H,
    NOP
    MPYF32   R4H,R1
    NOP
    SUBF32   R4H,R3H,
    NOP
    MPYF32   R1H,R1H,R4H
    NOP
    MPYF32   R0H,R0H,R1H
    MOV32    R0H,R1H
    MOV31    R4H,*--SP
    LRETR
```

Плав. тчк.
42 слова
31 такт

Модуль центрального процессора TMS320F28x

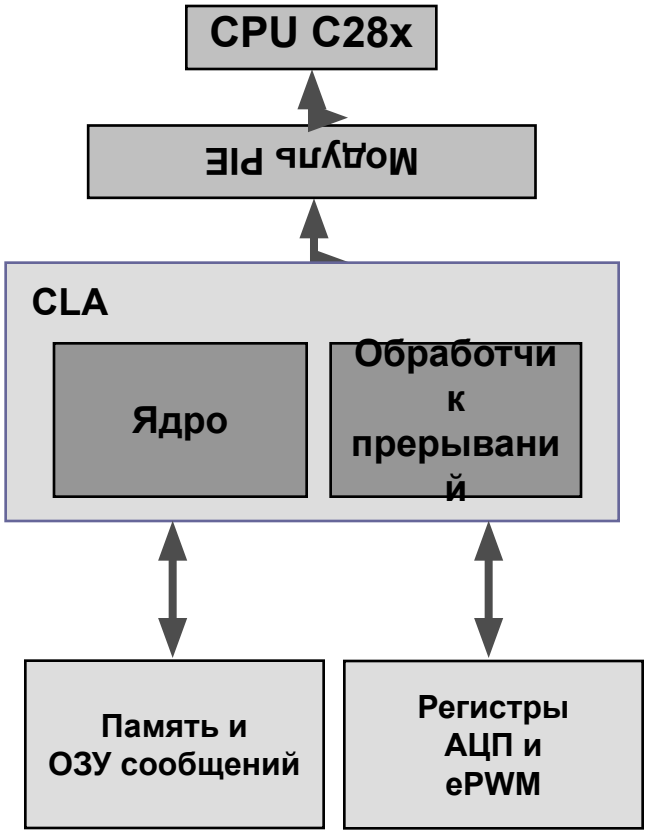
Эффективность использования серий Delfino:



Эквивалентная частота, например, 367 МГц означает, что разработчик, использующий C2812, должен был разогнать процессор до 367 МГц, чтобы получить производительность C28335.

Модуль центрального процессора TMS320F28x

Особенности модуля центрального процессора ЦСП серии Piccolo C2803x



Серия Piccolo C2803x содержит независимый 32-битный математический акселератор плавающей точки (CLA, Control Law Accelerator), который сокращает загрузку CPU, предоставляя возможность увеличения функционала.

- FPU-инструкции исполняются в полностью независимом конвейере (8 стадий);

- частота работы ЦСП серии C2803x – до 60 МГц;

- в CLA имеется независимый набор регистров, структура шин памяти и модулей обработки;

- алгоритмы исполняются CLA в параллель с CPU;

- CPU выделяет память программ/блоки памяти данных для CLA;

- имеется ОЗУ сообщений, которое служит для обмена данными между CPU и CLA;

- CLA имеет прямой доступ к регистрам АЦП и модуля расширенного ШИМ (ePWM);

- возможна генерация прерывания CPU от CLA.

CLA выполняет алгоритмы обратной связи эффективнее, разгружает CPU и увеличивает производительность. Сокращение требуемого числа тактов на исполнение математических операций – до 65%.

Модуль центрального процессора TMS320F28x

Без CLA
69 тактов

```
_ISR_BUCK:
        PUSH    AR1H:AR0H, XAR2, XAR3, XT
        SPM     0
        SETC    OVM,SXM
        MOVW    DP,#ADC_CTRL_REGS
        MOV     @ADCTRL2,#0x4100      ; Reset ADC sequencer back
        MOVW    DP,#PIE_REGS
        MOV     @PIEACK,#0xFFFF       ; Reset PIE
        MOVW    DP,#EPWM1_REGS
        MOV     @ETCLR1,#0x01         ; Clear EPWM1 Int flag
       MOVW    DP,#_Data3              ; DP   -> Buck 3 Data
        MOVL    XAR0,@_Ref3           ; XAR0 -> Buck 3 Ref 3
        MOVL    XAR1,#ADC_CH2         ; XAR1 -> ADC CH2 Regs (0-wait)
        MOVL    XAR2,#EPWM3_CMPA      ; XAR2 -> EPWM3, CMPA Regs
        MOVL    XAR3,@_Coef3          ; XAR3 -> Buck 3 Coef
        MOV     ACC,*XAR0++<<11       ; ACC = Ref3                    (Q15 -> Q26)
        SUB     ACC,*XAR1++<<14       ; ACC = e(n) = Ref3 - ADC_CH2   (Q12 -> Q26)
        MOVL    @_Data3_en_en1,ACC    ; Note: e(n) = e(n-1)           (Q26)
        MOVL    XT,ACC                ; XT  = e(n)                    (Q26)
        QMPYL   ACC,XT,*XAR3++        ; ACC = e(n)*B0                 (Q20 = Q26*Q26 - 32)
        ADDL    ACC,@_Data3_PostCalc  ; ACC = e(n)*B0 + PostCalc      (Q20 = Q20 + Q20)
        MINL    ACC,*XAR3++           ; Sat. to <= 0.9999..           (Q20)
        MAXL    ACC,*XAR3++           ; Sat. to >= 0.0000..           (Q20)
        LSL     ACC,#6                ; u(n) = ACC << 6               (Q20 -> Q26)
        MOVL    @_Data3_un_un1,ACC    ; Note: u(n) = u(n-1)           (Q26)
        LSL     ACC,#5                ; Duty = ACC << 5               (Q31)
        MOV     T,@AH                 ; T = Duty                      (Q31 -> Q15)
        MPYU    ACC,T,@_Data3_Period  ; ACC = Duty * Period           (Q16 = Q15 * Q1)
        MOV     T,@_Data3_ScaleFactor ; T = ScaleFactor               (Q8)
        MPYU    P,T,@AL               ; P = T * PL                    (Q24 = Q8 * Q16)
        MOVH    @AL,P                 ; AL = PH                       (Q24 -> Q8)
        ADD     ACC,#0x17F            ; Optimize by Round up          (AH=Q0, AL=Q8)
        MOVL    *XAR2,ACC             ; CMPA:CMPAHR(31:8) = ACC
        MOVL    XT,@_Data3_en2        ; XT   = e(n-2)                 (Q26)
        QMPYL   ACC,XT,*XAR3++        ; ACC  = e(n-2)*B2              (Q20 = Q26*Q26 - 32)
        MOVDL   XT,@_Data3_en1        ; XT   = e(n-1), e(n-2) = e(n-1) (Q26)
        QMPYL   P,XT,*XAR3++          ; P    = e(n-1)*B1              (Q20 = Q26*Q26 - 32)
        MOVL    XT,@_Data3_un2        ; XT   = u(n-2)                 (Q26)
        QMPYAL  P,XT,*XAR3++          ; P    = u(n-2)*A2,             (Q20 = Q26*Q26 -3 2)
        MOVDL   XT,@_Data3_un1        ; XT   = u(n-1), u(n-2) = u(n-1) (Q26)
        QMPYAL  P,XT,*XAR3            ; P    = u(n-1)*A1,             (Q20 = Q26*Q26 - 32)
        ADDL    ACC,@P                ; ACC += u(n-1)*A1              (Q20 = Q20+Q20)
        MOVL    @_Data3_PostCalc,ACC  ; Store value for next time     (Q20)
        POP     XT, XAR3, XAR2, AR1H:AR0H
        IRET
```

CLA
24 такта

```
_CLA_BUCK:
        MOV
        SUB     ACC,@
        MOVL    @_en1,ACC
        QMPYL   ACC,ACC,@_B0
        NOP
        ADDL    ACC,@_PostCalc
        MINL    ACC,#0.99999
        MAXL    ACC,#0.00000
        MOVL    @_un1,ACC << 6
        LSL     ACC,#12
        QMPYUL  ACC,ACC,@_Period
        MOVL    XT,@_B2
        MOVL    @CMPA:CMPAHR,ACC
        QMPYL   ACC,XT,@_en2
        MOVDL   XT,@_en1
        QMPYL   P,XT,@_B1
        MOVL    XT,@_un2
        QMPYAL  P,XT,@_A2
        MOVDL   XT,@_un1
        QMPYAL  P,XT,@_A1
        NOP
        ADDL    ACC,P
        MOVL    @_PostCalc,ACC << 0
        STOP
```

CLA выполняет алгоритм обратной связи эффективнее, разгружает ЦП и увеличивает производительность

Модуль центрального процессора TMS320F28x

Сравнение FPU (серии Delfino C2833x и C2834x) и CLA (серия Piccolo C2803x)

FPU	CLA
Общий с ядром C28x конвейер	Независимый конвейер (8 уровней)
Двухцикловые мат. операции	Одноцикловые мат. операции
8 результирующих регистров, общий с ядром C28x аккумулятор	4 результирующих регистра, 2 независимых аккумулятора
Поддержка стека, вложенных прерываний	Нет указателя стека и вложенных прерываний
Используется C28x переход, вызов и возврат	Собственные переход, вызов и возврат
Дополнительные инструкции для обмена данными между регистрами FPU и C28x	Дополнительные инструкции для передачи данных через ОЗУ
Целочисленные инструкции C28x	Собственные целочисленные инструкции: `AND, OR, XOR, ADD/SUB, Shift`
C/C++ или Ассемблер	Ассемблер
Один шаг сбрасывает весь конвейер	Один шаг сдвигает конвейер на цикл

Модуль центрального процессора TMS320F28x

Регистр - программный счетчик (PC) всегда указывает на команду, которая в настоящее время обрабатывается – команда, которая только достигла фазы декодирования D2 конвейера. Как только команда достигает этой фазы конвейера, ее выполнение не может быть прервано сигналами прерываний.

Счетчик программного возврата (RPC). Когда операция вызова подпрограммы с использованием команды `LCR` выполнена, адрес возврата сохраняется в регистре RPC, а старое значение RPC сохраняется в стеке (в двух 16-разрядных операциях). Когда операция возврата из подпрограммы `LRETR` выполнена, адрес возврата считывается из регистра RPC, а значение из стека записывается в регистр RPC (в двух 16-разрядных операциях). Другие команды вызова подпрограмм не используют регистр RPC.

Модуль центрального процессора TMS320F28x

C28x имеет два регистра состояния – ST0 и ST1, которые содержат различные биты флагов и служебные биты. Эти регистры могут быть сохранены и загружены из памяти, чтобы статус ЦСП был сохранен и восстановлен в процессе выполнения подпрограмм.

Регистры состояния организованы в соответствии с тем, как их содержимое изменяется в конвейере. Биты ST0 изменяются в фазе X конвейера; биты ST1 – в фазе D2.

Поразрядные поля регистра состояния ST0:

15 … 10	9 … 7	6	5	4	3	2	1	0
OVC/OVCU	PM	V	N	Z	C	TC	OVM	SXM
R/W-00 0000	R/W-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0

Модуль центрального процессора TMS320F28x

OVC/OVCU (биты $ST0_{10-15}$) – счетчик переполнений. Режим установки этих флагов активен, когда выключен режим переполнения (флаг OVM = 0).

Счетчик переполнений ведет себя неодинаково для знаковых и беззнаковых операций.

Для знаковых операций (OVC), счетчик переполнения – 6-разрядный знаковый счетчик с амплитудой от -32 до 31. Когда происходит переполнение ACC в положительном направлении (от 7FFF $FFFF_{16}$ до 8000 0000_{16}), OVC будет увеличен на 1. Когда происходит переполнение ACC в отрицательном направлении (от 8000 0000_{16} до 7FFF $FFFF_{16}$), OVC будет уменьшен на 1.

Для операций без знака (OVCU), счетчик будет инкрементирован при сложении, когда в аккумуляторе произошел перенос и декрементирован при вычитании, когда в аккумуляторе произошел заем.

При увеличении OVC после значения, равного 31, происходит его переполнение в значение -32. При уменьшении OVC из значения -32 происходит обратное переполнение в значение 31. При сбросе, OVC очищается. На OVC воздействуют переполнения только в регистре ACC, за исключением команд сравнения.

Модуль центрального процессора TMS320F28x

PM (бит $ST0_{7-9}$) – биты режима сдвига, задают сдвиговый режим выходных операций в регистре произведения P. Результат сдвига попадает в АЛУ или в память. После начальной установки все биты PM сброшены в 0.

V (бит $ST0_6$) – флаг переполнения. Если результат операции вызывает переполнение в регистре, хранящем результат, флаг V будет установлен и «защелкнут». Если переполнение не происходит, V не изменяется. Флаг V защелкнут, пока не будет очищен сбросом или командой условного перехода, которая проверяет V. Такой условный переход очищает V независимо от того, является ли проверенное условие (V = 0 или V = 1) истинным.

Переполнение происходит в ACC (и V установлен) если результат сложения или вычитания не может быть размещен в пределах диапазона знаковых чисел – от $8000\ 0000_{16}$ до $7FFF\ FFFF_{16}$.

Переполнение происходит в AH, AL, или другом 16-разрядном регистре или в ячейке памяти, если результат сложения или вычитания не может быть размещен в пределах от 8000_{16} до $7FFF_{16}$.

Команды `CMP`, `CMPB` и `CMPL` не воздействуют состояние флага V.

Модуль центрального процессора TMS320F28x

N (бит $ST0_5$) – флаг знака. N установлен, если результат операции – отрицательное число или сброшен, если результат – положительное число. После сброса N сброшен в 0. Если бит 31 ACC равен 0, ACC – положителен; если бит 31 равен 1, ACC отрицателен. Результат AH, AL, и других 16-разрядных регистров или данных в ячейках памяти также проверяются на отрицательное условие. Тогда значение бита 15 – знаковый разряд (1 указывает на отрицательное, 0 указывает на положительное число). Команда `TEST ACC` устанавливает флаг N, если значение в ACC отрицательно. Иначе команда сбрасывает флаг N.

Z ($ST0_4$) – флаг нуля. Z установлен, если результат некоторых операций – 0 или сброшен, если результат отличается от нуля. Это применяется к результатам, которые получены в ACC, AH, AL, другом регистре, или в ячейке памяти. После сброса, Z сброшен. Команда `TEST ACC` устанавливает Z, если значение в ACC – 0, иначе сбрасывает Z.

Модуль центрального процессора TMS320F28x

C (бит $ST0_3$) – флаг переноса. Этот флаг показывает, когда сложение, инкремент генерируют перенос, или когда вычитание, сравнение, декремент генерируют заем. Этот флаг также устанавливают операции программного сдвига ACC (команды `ROR`, `ROL`) и аппаратные сдвиги (barrel shift) в ACC, AH, и AL. В результате сложения/инкремента, C будет установлен, если генерируется перенос; иначе C будет сброшен. Имеется одно исключение: если используется команда `ADD` со сдвигом 16 (`ADD ACC,loc16<<shift`), C может устанавливаться, но не может сбрасываться.

В результате вычитания/декремента/сравнения, C будет сброшен, если вычитание генерирует перенос; иначе C будет установлен. Имеется одно исключение: если используется команда `SUB` со сдвигом 16 (`SUB ACC,loc16<<shift`), C может сбрасываться, но не может устанавливаться.

Этот бит может быть индивидуально установлен и очищен командами `SETC C` и `CLRC C` соответственно. После начального сброса, C сброшен в 0.

Модуль центрального процессора TMS320F28x

TC (бит $ST0_2$) – флаг тест/управление. Этот бит показывает результат тестирования, выполненного любой TBIT-командой (тест-бит) или командой NORM (нормализация).

Команда TBIT проверяет выбранный бит. Когда команда TBIT выполнена, флаг TC установлен, если проверяемый бит – 1 или сброшен, если проверяемый бит – 0.

```
; if( VarA.Bit4 = 1 )
;     VarB.Bit6 = 1;
; else
;     VarB.Bit6 = 0;
   TBIT   @VarA,#4       ; Test bit 4 of VarA contents
   SB     $10,NTC        ; Branch if TC = 0
   TSET   @VarB,#6       ; Set bit 6 of VarB contents
   SB     $20,UNC        ; Branch unconditionally
$10:                     ;
   TCLR   @VarB,#6       ; Clear bit 6 of VarB contents
$20:                     ;
```

Когда команда NORM выполнена, TC изменяется следующим образом: Если ACC содержит 0, TC установлен. Если содержимое ACC отличается от 0, CPU вычисляет исключающее ИЛИ битов 31 и 30 ACC, и затем загружает TC результатом.

Этот бит может быть индивидуально установлен или сброшен командой SETC TC или CLRC TC соответственно. После сброса, TC сброшен в 0.

Модуль центрального процессора TMS320F28x

OVM (бит $ST0_1$) – флаг режима переполнения. Когда ACC принимает результат сложения или вычитания, и результат вызывает переполнение, OVM определяет, как CPU обрабатывает переполнение:

0 – нормальное переполнение результата в ACC. Состояние флагов OVC отражает переполнение.

1 – состояние флагов OVC не изменяется, а ACC заполняется максимально возможным положительным или отрицательным значением следующим образом:

- если ACC переполняется в положительном направлении (от 7FFF $FFFF_{16}$ до 8000 0000_{16}), ACC заполняется значением 7FFF $FFFF_{16}$.

- если ACC переполняется в отрицательном направлении (от 8000 0000_{16} до 7FFF $FFFF_{16}$), ACC заполняется значением 8000 0000_{16}.

Этот бит может быть индивидуально установлен и сброшен соответственно командами `SETC OVM` и `CLRC OVM`. После начального сброса OVM сброшен.

Модуль центрального процессора TMS320F28x

SXM (бит $ST0_0$) – флаг режима расширения знака. На флаг SXM воздействуют команды `MOV`, `ADD` и `SUB`, которые используют 16-битные операции в 32-разрядном аккумуляторе. Когда 16-разрядное значение загружено (`MOV`), добавлено (`ADD`) или вычтено (`SUB`) из ACC, SXM определяет режим обработки значения со знаком, расширенным в течение операции следующим образом:

0 – расширение знака подавлено (значение будет обрабатываться как беззнаковое).

1 – расширение знака допускается (значение будет обрабатываться как знаковое).

Этот флаг может быть индивидуально установлен и очищен командой `SETC SXM` и командой `CLRC SXM`, соответственно. После начального сброса DSP флаг SXM сброшен.

Модуль центрального процессора TMS320F28x

Поразрядные поля регистра состояния ST1:

15 … 13	12	11	10	9	8
ARP	XF	M0M1MAP	Reserved	OBJMODE	AMODE
R/W-000	R/W-0	R/W-1	R/W-0	R/W-0	R/W-0

7	6	5	4	3	2	1	0
IDLESTAT	EALLOW	LOOP	SPA	VMAP	PAGE0	DBGM	INTM
R-0	R/W-0	R-0	R/W-0	R/W-1	R/W-0	R/W-1	R/W-1

Модуль центрального процессора TMS320F28x

ARP (биты $ST1_{15-13}$) – 3-битный указатель текущего вспомогательного регистра XAR0..XAR7.

Значение ARP	*Выбранный дополнительный регистр*
000	XAR0 (выбран после начальной установки ЦСП)
001	XAR1
010	XAR2
011	XAR3
100	XAR4
101	XAR5
110	XAR6
111	XAR7

После сброса DSP указатель ARP установлен в 000.

Модуль центрального процессора TMS320F28x

XF (бит $ST1_{12}$) – флаг, отражающий текущее состояние вывода /XF_XPLLDIS.

Программная установка флага – `SETC XF`;

сброс – `CLRC XF`.

При использовании этих команд конвейер выполнения команд не может быть прерван. Бит XF (в составе регистра ST1) сохраняется и восстанавливается при обработке прерываний.

M0M1MAP (бит $ST1_{11}$) – флаг режима карты памяти. Он всегда равен 1 в объектном режиме C28x (это значение флаг имеет после начальной установки DSP). Когда необходимо использовать C27x-совместимый режим, этот флаг может быть установлен в 0. При этом адреса областей памяти M0 и M1 меняются местами (только в памяти программ, но не в памяти данных) и указатель стека по умолчанию имеет значение 0x000.

Бит $ST1_{10}$ – резервный бит.

OBJMODE (бит $ST1_9$) – флаг режима объектной совместимости (0 для C27x-режима и 1 для C28x-режима).

Программная установка флага – команды `SETC OBJMODE; C28OBJ`;

сброс – `CLRC OBJMODE; C27OBJ`.

Бит OBJMODE (в составе регистра ST1) сохраняется и восстанавливается при обработке прерываний. После начальной установки ЦСП флаг имеет нулевое значение.

Модуль центрального процессора TMS320F28x

AMODE (бит $ST1_8$) – флаг режима адресации. Этот бит, в сочетании с битом PAGE0 используется для выбора соответствующего режима адресации:

AMODE=0 – в режиме прямой адресации DP дополняется 6-битным смещением, и некоторые режимы косвенной адресации не поддерживаются (C28x-режим);

AMODE=1 – в режиме прямой адресации DP дополняется 7-битным смещением, и поддерживаются все режимы косвенной адресации.

Программная установка флага – команды `SETC AMODE; LPADDR;`

сброс – `CLRC AMODE; C28ADDR`.

При использовании этих команд конвейер выполнения команд не может быть прерван. Бит AMODE (в составе регистра ST1) сохраняется и восстанавливается при обработке прерываний. После начальной установки DSP флаг имеет нулевое значение.

IDLESTAT (бит $ST1_7$) – флаг-индикатор выполненной инструкции `IDLE`. Доступен только по чтению. Флаг может быть сброшен по факту обслуживания прерывания и после начального сброса ЦСП. После обслуживания прерывания значение бита IDLESTAT из стека не восстанавливается.