

**Memory access instructions**

```

LDR    Rd, [Rn]           ; load 32-bit number at [Rn] to Rd
LDR    Rd, [Rn,#off]      ; load 32-bit number at [Rn+off] to Rd
LDR    Rd, =value         ; set Rd equal to any 32-bit value (PC rel)
LDRH   Rd, [Rn]           ; load unsigned 16-bit at [Rn] to Rd
LDRH   Rd, [Rn,#off]      ; load unsigned 16-bit at [Rn+off] to Rd
LDRSH  Rd, [Rn]           ; load signed 16-bit at [Rn] to Rd
LDRSH  Rd, [Rn,#off]      ; load signed 16-bit at [Rn+off] to Rd
LDRB   Rd, [Rn]           ; load unsigned 8-bit at [Rn] to Rd
LDRB   Rd, [Rn,#off]      ; load unsigned 8-bit at [Rn+off] to Rd
LDRSB  Rd, [Rn]           ; load signed 8-bit at [Rn] to Rd
LDRSB  Rd, [Rn,#off]      ; load signed 8-bit at [Rn+off] to Rd
STR     Rt, [Rn]          ; store 32-bit Rt to [Rn]
STR     Rt, [Rn,#off]     ; store 32-bit Rt to [Rn+off]
STRH    Rt, [Rn]          ; store least sig. 16-bit Rt to [Rn]
STRH    Rt, [Rn,#off]     ; store least sig. 16-bit Rt to [Rn+off]
STRB    Rt, [Rn]          ; store least sig. 8-bit Rt to [Rn]
STRB    Rt, [Rn,#off]     ; store least sig. 8-bit Rt to [Rn+off]
PUSH    {Rt}              ; push 32-bit Rt onto stack
POP     {Rd}              ; pop 32-bit number from stack into Rd
ADR     Rd, label         ; set Rd equal to the address at label
MOV{S}  Rd, <op2>         ; set Rd equal to op2
MOV     Rd, #iml6         ; set Rd equal to iml6, iml6 is 0 to 65535
MVN{S}  Rd, <op2>         ; set Rd equal to -op2

```

**Branch instructions**

```

B       label      ; branch to label      Always
BEQ     label      ; branch if Z == 1      Equal
BNE     label      ; branch if Z == 0      Not equal
BCS     label      ; branch if C == 1      Higher or same, unsigned ≥
BHS     label      ; branch if C == 1      Higher or same, unsigned ≥
BCC     label      ; branch if C == 0      Lower, unsigned <
BLO     label      ; branch if C == 0      Lower, unsigned <
BMI     label      ; branch if N == 1      Negative
BPL     label      ; branch if N == 0      Positive or zero
BVS     label      ; branch if V == 1      Overflow
BVC     label      ; branch if V == 0      No overflow
BHI     label      ; branch if C==1 and Z==0 Higher, unsigned >
BLS     label      ; branch if C==0 or Z==1 Lower or same, unsigned ≤
BGE     label      ; branch if N == V      Greater than or equal, signed ≥
BLT     label      ; branch if N != V      Less than, signed <
BGT     label      ; branch if Z==0 and N==V Greater than, signed >
BLE     label      ; branch if Z==1 or N!=V Less than or equal, signed ≤
BX      Rm         ; branch indirect to location specified by Rm
BL      label      ; branch to subroutine at label
BLX     Rm         ; branch to subroutine indirect specified by Rm

```

**Interrupt instructions**

```

CPSIE   I           ; enable interrupts (I=0)
CPSID   I           ; disable interrupts (I=1)

```

**Logical instructions**

```

AND{S}  {Rd}, {Rn}, <op2> ; Rd=Rn&op2      (op2 is 32 bits)
ORR{S}  {Rd}, {Rn}, <op2> ; Rd=Rn|op2      (op2 is 32 bits)
EOR{S}  {Rd}, {Rn}, <op2> ; Rd=Rn^op2      (op2 is 32 bits)
BIC{S}  {Rd}, {Rn}, <op2> ; Rd=Rn&(~op2)   (op2 is 32 bits)
ORN{S}  {Rd}, {Rn}, <op2> ; Rd=Rn|(~op2)   (op2 is 32 bits)
LSR{S}  Rd, Rm, Rs      ; logical shift right Rd=Rm>>Rs (unsigned)

```

```

LSR{S} Rd, Rm, #n      ; logical shift right Rd=Rm>>n (unsigned)
ASR{S} Rd, Rm, Rs      ; arithmetic shift right Rd=Rm>>Rs (signed)
ASR{S} Rd, Rm, #n      ; arithmetic shift right Rd=Rm>>n (signed)
LSL{S} Rd, Rm, Rs      ; shift left Rd=Rm<<Rs (signed, unsigned)
LSL{S} Rd, Rm, #n      ; shift left Rd=Rm<<n (signed, unsigned)

```

**Arithmetic instructions**

```

ADD{S} {Rd,} Rn, <op2> ; Rd = Rn + op2
ADD{S} {Rd,} Rn, #im12 ; Rd = Rn + im12, im12 is 0 to 4095
SUB{S} {Rd,} Rn, <op2> ; Rd = Rn - op2
SUB{S} {Rd,} Rn, #im12 ; Rd = Rn - im12, im12 is 0 to 4095
RSB{S} {Rd,} Rn, <op2> ; Rd = op2 - Rn
RSB{S} {Rd,} Rn, #im12 ; Rd = im12 - Rn
CMP   Rn, <op2>        ; Rn - op2      sets the NZVC bits
CMN   Rn, <op2>        ; Rn - (-op2)   sets the NZVC bits
MUL{S} {Rd,} Rn, Rm     ; Rd = Rn * Rm   signed or unsigned
MLA   Rd, Rn, Rm, Ra     ; Rd = Ra + Rn*Rm signed or unsigned
MLS   Rd, Rn, Rm, Ra     ; Rd = Ra - Rn*Rm signed or unsigned
UDIV  {Rd,} Rn, Rm       ; Rd = Rn/Rm    unsigned
SDIV  {Rd,} Rn, Rm       ; Rd = Rn/Rm    signed

```

**Notes** Ra Rd Rm Rn Rt represent 32-bit registers

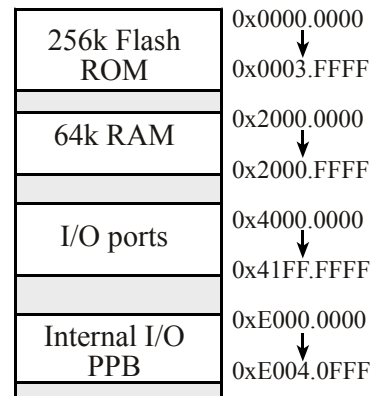
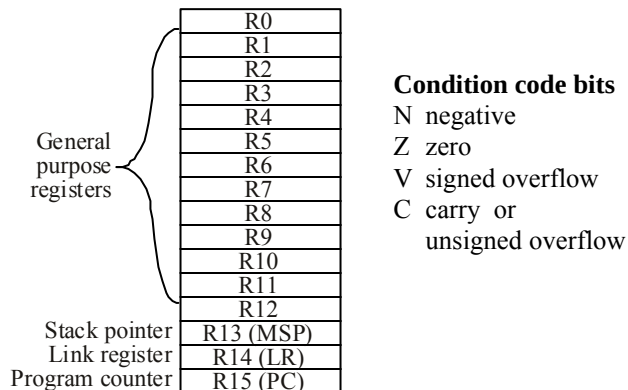
value any 32-bit value: signed, unsigned, or address  
 {S} if S is present, instruction will set condition codes  
 #im12 any value from 0 to 4095  
 #im16 any value from 0 to 65535  
 {Rd,} if Rd is present Rd is destination, otherwise Rn  
 #n any value from 0 to 31  
 #off any value from -255 to 4095  
 label any address within the ROM of the microcontroller  
 op2 the value generated by <op2>

Examples of flexible operand <op2> creating the 32-bit number. E.g., Rd = Rn+op2

```

ADD Rd, Rn, Rm      ; op2 = Rm
ADD Rd, Rn, Rm, LSL #n ; op2 = Rm<<n Rm is signed, unsigned
ADD Rd, Rn, Rm, LSR #n ; op2 = Rm>>n Rm is unsigned
ADD Rd, Rn, Rm, ASR #n ; op2 = Rm>>n Rm is signed
ADD Rd, Rn, #constant ; op2 = constant, where X and Y are hexadecimal digits:
    • produced by shifting an 8-bit unsigned value left by any number of bits
    • in the form 0x00XY00XY
    • in the form 0xXY00XY00
    • in the form 0xXYXYXYXY

```



DCB 1,2,3 ; allocates three 8-bit byte(s)  
 DCW 1,2,3 ; allocates three 16-bit halfwords

DCD 1,2,3 ; allocates three 32-bit words  
 SPACE 4 ; reserves 4 bytes

| Address     | 7    | 6    | 5     | 4     | 3     | 2     | 1     | 0     | Name               |
|-------------|------|------|-------|-------|-------|-------|-------|-------|--------------------|
| \$400F.E108 |      |      | GPIOF | GPIOE | GPIOD | GPIOC | GPIOB | GPIOA | SYSCTL_RCGC2_R     |
| \$4000.43FC | DATA | DATA | DATA  | DATA  | DATA  | DATA  | DATA  | DATA  | GPIO_PORTA_DATA_R  |
| \$4000.4400 | DIR  | DIR  | DIR   | DIR   | DIR   | DIR   | DIR   | DIR   | GPIO_PORTA_DIR_R   |
| \$4000.4420 | SEL  | SEL  | SEL   | SEL   | SEL   | SEL   | SEL   | SEL   | GPIO_PORTA_AFSEL_R |
| \$4000.451C | DEN  | DEN  | DEN   | DEN   | DEN   | DEN   | DEN   | DEN   | GPIO_PORTA_DEN_R   |

Table 4.5. Some TM4C123/LM4F120 parallel ports. Each register is 32 bits wide. Bits 31 – 8 are zero.

| Address    | 31 | 30 | 29-7 | 6     | 5     | 4 | 3 | 2 | 1 | 0 | Name       |
|------------|----|----|------|-------|-------|---|---|---|---|---|------------|
| 0xE000E100 |    | F  | ...  | UART1 | UART0 | E | D | C | B | A | NVIC_EN0_R |

| Address    | 31-24 | 23-17                                   | 16    | 15-3 | 2       | 1     | 0      | Name              |
|------------|-------|-----------------------------------------|-------|------|---------|-------|--------|-------------------|
| \$E000E010 | 0     | 0                                       | COUNT | 0    | CLK_SRC | INTEN | ENABLE | NVIC_ST_CTRL_R    |
| \$E000E014 | 0     | 24-bit RELOAD value                     |       |      |         |       |        | NVIC_ST_RELOAD_R  |
| \$E000E018 | 0     | 24-bit CURRENT value of SysTick counter |       |      |         |       |        | NVIC_ST_CURRENT_R |

| Address    | 31-29   | 28-24 | 23-21  | 20-8 | 7-5   | 4-0 | Name            |
|------------|---------|-------|--------|------|-------|-----|-----------------|
| \$E000ED20 | SYSTICK | 0     | PENDSV | 0    | DEBUG | 0   | NVIC_SYS_PRI3_R |

Table 9.6. SysTick registers.

Table 9.6 shows the SysTick registers used to create a periodic interrupt. SysTick has a 24-bit counter that decrements at the bus clock frequency. Let  $f_{BUS}$  be the frequency of the bus clock, and let  $n$  be the value of the **RELOAD** register. The frequency of the periodic interrupt will be  $f_{BUS}/(n+1)$ . First, we clear the **ENABLE** bit to turn off SysTick during initialization. Second, we set the **RELOAD** register. Third, we write to the **NVIC\_ST\_CURRENT\_R** value to clear the counter. Lastly, we write the desired mode to the control register, **NVIC\_ST\_CTRL\_R**. To turn on the SysTick, we set the **ENABLE** bit. We must set **CLK\_SRC**=1, because **CLK\_SRC**=0 external clock mode is not implemented on the LM3S/LM4F family. We set **INTEN** to enable interrupts. The standard name for the SysTick ISR is **SysTick\_Handler**.

| Address     | 31-17 | 16    | 15-10 | 9           | 8     | 7-0   | Name           |              |             |
|-------------|-------|-------|-------|-------------|-------|-------|----------------|--------------|-------------|
| \$400F.E000 |       | ADC   |       | MAXADCS     |       | PD    | SYSCTL_RCGC0_R |              |             |
|             | 31-14 | 13-12 | 11-10 | 9-8         | 7-6   | 5-4   | 3-2            | 1-0          |             |
| \$4003.8020 |       | SS3   |       | SS2         |       | SS1   |                | SS0          | ADC_SSPRI_R |
|             | 31-16 |       |       | 15-12       |       | 11-8  | 7-4            | 3-0          |             |
| \$4003.8014 |       |       |       | EM3         |       | EM2   | EM1            | EM0          | ADC_EMUX_R  |
|             | 31-4  |       |       | 3           | 2     | 1     | 0              |              |             |
| \$4003.8000 |       |       |       | ASEN3       | ASEN2 | ASEN1 | ASEN0          | ADC_ACTSS_R  |             |
| \$4003.80A0 |       |       |       | MUX0        |       |       |                | ADC_SSMUX3_R |             |
| \$4003.80A4 |       |       |       | TS0         | IE0   | END0  | D0             | ADC_SSCTL3_R |             |
| \$4003.8028 |       |       |       | SS3         | SS2   | SS1   | SS0            | ADC_PSSI_R   |             |
| \$4003.8004 |       |       |       | INR3        | INR2  | INR1  | INR0           | ADC_RIS_R    |             |
| \$4003.8008 |       |       |       | MASK3       | MASK2 | MASK1 | MASK0          | ADC_IM_R     |             |
| \$4003.800C |       |       |       | IN3         | IN2   | IN1   | IN0            | ADC_ISC_R    |             |
|             | 31-10 |       |       |             | 11-0  |       |                |              |             |
| \$4003.80A8 |       |       |       | 12-bit DATA |       |       |                | ADC_SSFIFO3  |             |

Table 10.3. The TM4C123/LM4F120ADC registers. Each register is 32 bits wide.

Set MAXADCSPPD to 00 for slow speed operation. The ADC has four sequencers, but we will use only sequencer 3. We set the **ADC\_SSPRI\_R** register to 0x3210 to make sequencer 3 the lowest priority. Because we are using just one sequencer, we just need to make sure each sequencer has a unique priority. We set bits 15–12 (**EM3**) in the **ADC\_EMUX\_R** register to specify how the ADC will be triggered. If we specify software start (**EM3**=0x0), then the software writes an 8 (**SS3**) to the **ADC\_PSSI\_R** to initiate a conversion on sequencer 3. Bit 3 (**INR3**) in the **ADC\_RIS\_R** register will be set when the conversion is complete. We can enable and disable the sequencers using the **ADC\_ACTSS\_R**

register. There are 11 on the TM4C123/LM4F120. Which channel we sample is configured by writing to the **ADC\_SSMUX3\_R** register. The **ADC\_SSCTL3\_R** register specifies the mode of the ADC sample. Clear **TS0**. We set **IE0** so that the **INR3** bit is set on ADC conversion, and clear it when no flags are needed. We will set **IE0** for both interrupt and busy-wait synchronization. When using sequencer 3, there is only one sample, so **END0** will always be set, signifying this sample is the end of the sequence. Clear the **D0** bit. The **ADC\_RIS\_R** register has flags that are set when the conversion is complete, assuming the **IE0** bit is set. Do not set bits in the **ADC\_IM\_R** register because we do not want interrupts.

UART0 pins are on PA1 (transmit) and PA0 (receive). The **UART0\_IBRD\_R** and **UART0\_FBRD\_R** registers specify the baud rate. The baud rate **divider** is a 22-bit binary fixed-point value with a resolution of  $2^{-6}$ . The **Baud16** clock is created from the system bus clock, with a frequency of (Bus clock frequency)/**divider**. The baud rate is

$$\text{Baud rate} = \text{Baud16}/16 = (\text{Bus clock frequency})/(16 * \text{divider})$$

We set bit 4 of the **UART0\_LCRH\_R** to enable the hardware FIFOs. We set both bits 5 and 6 of the **UART0\_LCRH\_R** to establish an 8-bit data frame. The **RTRIS** is set on a receiver timeout, which is when the receiver FIFO is not empty and no incoming frames have occurred in a 32-bit time period. The arm bits are in the **UART0\_IM\_R** register. To acknowledge an interrupt (make the trigger flag become zero), software writes a 1 to the corresponding bit in the **UART0\_IC\_R** register.

We set bit 0 of the **UART0\_CTL\_R** to enable the UART. Writing to **UART0\_DR\_R** register will output on the UART. This data is placed in a 16-deep transmit hardware FIFO. Data are transmitted first come first serve. Received data are place in a 16-deep receive hardware FIFO. Reading from **UART0\_DR\_R** register will get one data from the receive hardware FIFO. The status of the two FIFOs can be seen in the **UART0\_FR\_R** register (FF is FIFO full, FE is FIFO empty). The standard name for the UART0 ISR is **UART0\_Handler**. RXIFLSEL specifies the receive FIFO level that causes an interrupt (010 means interrupt on  $\geq \frac{1}{2}$  full, or 7 to 8 characters). TXIFLSEL specifies the transmit FIFO level that causes an interrupt (010 means interrupt on  $\leq \frac{1}{2}$  full, or 9 to 8 characters).

|             |       |       |       |          |         |       |            |             |              |              |
|-------------|-------|-------|-------|----------|---------|-------|------------|-------------|--------------|--------------|
|             | 31-12 | 11    | 10    | 9        | 8       | 7-0   | Name       |             |              |              |
| \$4000.C000 |       | OE    | BE    | PE       | FE      | DATA  | UART0_DR_R |             |              |              |
|             | 31-3  |       |       | 3        | 2       | 1     | 0          |             |              |              |
| \$4000.C004 |       |       |       | OE       | BE      | PE    | FE         | UART0_RSR_R |              |              |
|             | 31-8  | 7     | 6     | 5        | 4       | 3     | 2-0        |             |              |              |
| \$4000.C018 |       | TXFE  | RXFF  | TXFF     | RXFE    | BUSY  |            |             | UART0_FR_R   |              |
|             | 31-16 |       |       |          | 15-0    |       |            |             |              |              |
| \$4000.C024 |       |       |       |          | DIVINT  |       |            |             | UART0_IBRD_R |              |
|             | 31-6  |       |       |          | 5-0     |       |            |             |              |              |
| \$4000.C028 |       |       |       |          | DIVFRAC |       |            |             | UART0_FBRD_R |              |
|             | 31-8  | 7     | 6-5   | 4        | 3       | 2     | 1          | 0           |              |              |
| \$4000.C02C |       | SPS   | WPEN  | FEN      | STP2    | EPS   | PEN        | BRK         | UART0_LCRH_R |              |
|             | 31-10 | 9     | 8     | 7        | 6-3     | 2     | 1          | 0           |              |              |
| \$4000.C030 |       | RXE   | TXE   | LBE      |         | SIRLP | SIREN      | UARTEN      | UART0_CTL_R  |              |
|             | 31-6  |       |       | 5-3      |         |       | 2-0        |             |              |              |
| \$4000.C034 |       |       |       | RXIFLSEL |         |       | TXIFLSEL   |             |              | UART0_IFLS_R |
|             | 31-11 | 10    | 9     | 8        | 7       | 6     | 5          | 4           |              |              |
| \$4000.C038 |       | OEIM  | BEIM  | PEIM     | FEIM    | RTIM  | TXIM       | RXIM        |              | UART0_IM_R   |
| \$4000.C03C |       | OERIS | BERIS | PERIS    | FERIS   | RTRIS | TXRIS      | RXRIS       |              | UART0_RIS_R  |
| \$4000.C040 |       | OEMIS | BEMIS | PEMIS    | FEMIS   | RTMIS | TXMIS      | RXMIS       |              | UART0_MIS_R  |
| \$4000.C044 |       | OEIC  | BEIC  | PEIC     | FEIC    | RTIC  | TXIC       | RXIC        |              | UART0_IC_R   |

Table 11.2. UART0 registers. Each register is 32 bits wide. Shaded bits are zero.