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, #im16      ; set Rd equal to im16, im16 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\gg\#n (unsigned)

ASR\{S\} Rd, Rm, Rs ; arithmetic shift right Rd=Rm\gg\#Rs (signed)

ASR\{S\} Rd, Rm, \#n ; arithmetic shift right Rd=Rm\gg\#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, \langle op2\rangle ; Rd = Rn + op2

ADD\{S\} \{Rd,\} Rn, \#im12 ; Rd = Rn + im12, im12 is 0 to 4095

SUB\{S\} \{Rd,\} Rn, \langle op2\rangle ; Rd = Rn - op2

SUB\{S\} \{Rd,\} Rn, \#im12 ; Rd = Rn - im12, im12 is 0 to 4095

RSB\{S\} \{Rd,\} Rn, \langle op2\rangle ; Rd = op2 - Rn

RSB\{S\} \{Rd,\} Rn, \#im12 ; Rd = im12 - Rn

CMP Rn, \langle op2\rangle ; Rn - op2 sets the NZVC bits

CMN Rn, \langle op2\rangle ; Rn - (-op2) sets the NZVC bits

MUL\{S\} \{Rd,\} Rn, Rm ; Rd = Rn \ast 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 \langle op2\rangle

### Examples of flexible operand \langle op2\rangle 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\gg\#n Rm is unsigned

ADD Rd, Rn, Rm, ASR \#n ; op2 = Rm\gg\#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

### Stack pointer

### Link register

### Program counter

### General purpose registers

### DCB 1,2,3 ; allocates three 8-bit byte(s)

### DCW 1,2,3 ; allocates three 16-bit halfwords

### Condition code bits

N negative

Z zero

V signed overflow

C carry or unsigned overflow

### 256k Flash ROM

256k Flash ROM

0x0000.0000

0x0003.FFFF

0x0200.0000

0x0200.0000

0x4000.0000

0x4000.0000

0xE000.0000

0xE000.0000

## I/O ports

I/O ports

0x41FF.FFFF

## Internal I/O

Internal I/O

0xE000.0000

0xE004.0FFF

## PPB

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

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

<table>
<thead>
<tr>
<th>Address</th>
<th>31</th>
<th>30</th>
<th>29-7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>0xE000E100</td>
<td>F</td>
<td>...</td>
<td>UART1</td>
<td>E</td>
<td>D</td>
<td>C</td>
<td>B</td>
<td>A</td>
<td></td>
<td>NVIC_EN0_R</td>
<td></td>
</tr>
</tbody>
</table>

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.

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

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

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

\[ \text{Baud rate} = \frac{\text{Baud16}}{16} = \frac{(\text{Bus clock frequency})}{(16 \times \text{divider})} \]

We set bit 4 of the \texttt{UART0\_LCRH\_R} to enable the hardware FIFOs. We set both bits 5 and 6 of the \texttt{UART0\_LCRH\_R} to establish an 8-bit data frame. The \texttt{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 \texttt{UART0\_IM\_R} register. To acknowledge an interrupt (make the trigger flag become zero), software writes a 1 to the corresponding bit in the \texttt{UART0\_IC\_R} register. We set bit 0 of the \texttt{UART0\_CTL\_R} to enable the UART. Writing to \texttt{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 placed in a 16-deep receive hardware FIFO. Reading from \texttt{UART0\_DR\_R} register will get one data from the receive hardware FIFO. The status of the two FIFOs can be seen in the \texttt{UART0\_FR\_R} register (FF is FIFO full, FE is FIFO empty). The standard name for the UART0 ISR is \texttt{UART0\_Handler}. \texttt{RXIFLSEL} specifies the receive FIFO level that causes an interrupt (010 means interrupt on ≥½ full, or 7 to 8 characters). \texttt{TXIFLSEL} specifies the transmit FIFO level that causes an interrupt (010 means interrupt on ≤½ full, or 9 to 8 characters).

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

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