Inhalt

Block 1: MC-Systemteile 2
Block 2: HW- und SW-Entwicklungsumgebung 4
Block 3: Assembler & Einführung HCS08 5
Block 4: Assemblerdirektiven & Adressierungsarten 6
Block 5: Adressierungsarten & Programmiertechniken 7
Block 6: Weitere HCS08-Befehle & Assemblerprogrammierung 8
Block 7: Unterprogramme & Stack 9
Block 8: Interruptsystem & Timer (Teil 1) 10
Block 9/10: Output-Compare & Input-Capture 11
Block 11: Pulse-Width Modulation (PWM) 12
Block 12/13: IIC-Bus 13
Block 14: A/D-Wandler 14
Block 15/16: RS232 Serielle Kommunikation 15
Block 17/18: RTOS 16

Block 1: MC-Systemteile

1. Stellen Sie -4 im Zweier-Komplement als 16-bit Hex-Zahl dar.

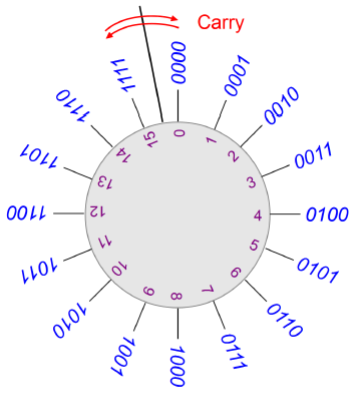
FF FC Binär: 1111 1111 1111 1100

Zweier-Komplement (4-Bit): -8 … 7

Zweier-Komplement (8-Bit): -128 … 127

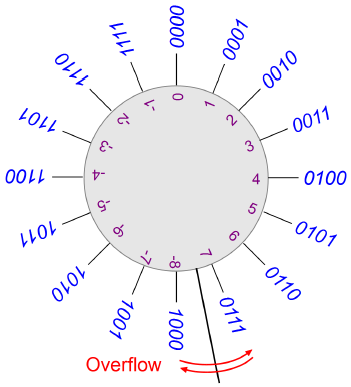
Zweier-Komplement (16-Bit): - 32‘768 … 32‘767

Vorzeichenlose Zahlen (unsigned)



Übertrag (**Carry**) beim Übergang zwischen kleinster und grösster Zahl

Zahlen mit Vorzeichen (signed)



Überlauf (**Overflow**) beim Übergang zwischen betragsmässig grössten Zahlen

Binärwert	Vorzeichenlose Zahl	Zweier-Komplement
0000	0	0
0001	1	1
0010	2	2
0011	3	3
0100	4	4
0101	5	5
0110	6	6
0111	7	7
1000	8	**–8**
1001	9	**–7**
1010	10	**–6**
1011	11	**–5**
1100	12	**–4**
1101	13	**–3**
1110	14	**–2**
1111	15	**–1**

Online Rechner: http://manderc.manderby.com/concepts/umrechner/index.php

2. Wie viele Bit-Speicherplätze beinhaltet ein 32 K x 8 Speicher? (Angabe in Hex)

1K = 1024-Bit

*32K * 8 = 32* 1024 * 8 = 262‘144 → 0x4 0000 Bit-Speicherplätze*

Wie viele Adressleitungen besitzt der Speicher?

8

Wie lautet die höchste Adresse?

0x3 FFFF (262‘143)

3. Worin besteht der Unterschied zwischen einem Mikrocontroller und einem Mikroprozessor?

Ein Mikroprozessor ist ein Prozessor in sehr kleinem Massstab, bei dem alle Bausteine des Prozessors auf einem Mikrochip vereinigt sind. (Integrated Circuit, kurz IC).

Als Mikrocontroller (auch µController, µC, MCU) werden Halbleiterchips bezeichnet, die mit dem Prozessor auch Peripheriefunktionen (Ein-/Ausgänge) auf einem Chip vereinen. In vielen Fällen befindet sich der Arbeits- und Programmspeicher ebenfalls teilweise oder komplett auf demselben Chip. Ein Mikrocontroller ist praktisch ein Ein-Chip-Computersystem. Für manche Mikrocontroller wird auch der Begriff System on a Chip oder SoC verwendet.

4. Was sind die 3 Haupt-Systemteile eines jeden Mikrocontrollers?

CPU, Speicher und Ein-/Ausgänge (&Peripherie)

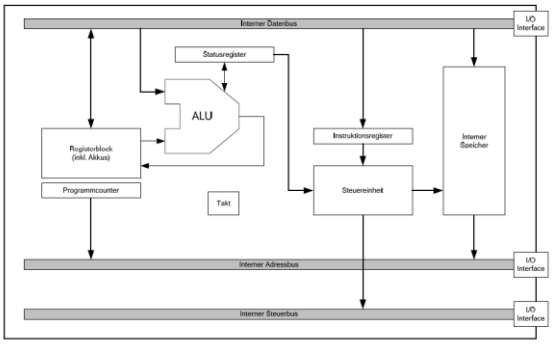
5. Welche Busse unterscheidet man in einer MCU?

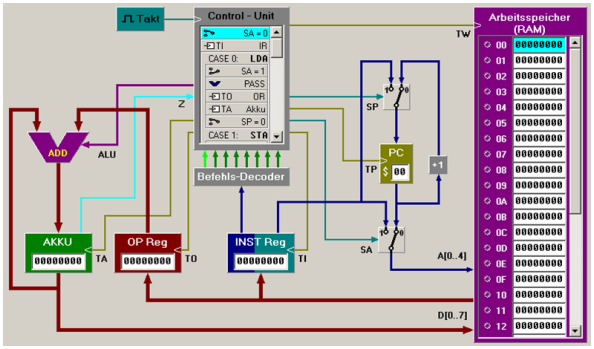
Steuerbus, Datenbus, Adressbus

Welche der Busse sind bidirektional?

Steuerbus und Datenbus

6. Welche Funktionseinheiten beinhaltet eine CPU?





- *Steuerlogik*
- *Befehlsregister*
- *ALU (Arithmetical Logical Unit)*
- *CPU-Register (Akku, CCR, Index- / Adressregister)*
- *Stackpointer*
- *Programcounter*
- *Busse (Daten-, Adress-, Steuerbus)*

7. Welche Schritte beinhaltet der Befehlszyklus?

Fetch *Befehlswort aus dem Speicher ins Befehlsregister der CPU holen*

Decode *Befehl erkennen, zur Festlegung der Mikro-Programmschritte im Steuerwerk*

Execute *Befehlsausführung, Abarbeitung des Mikro-Programms durch das Steuerwerk*

8. Welche Arten von Registern existieren in einer MCU?

- *Index-Register*
- *Statusregister CCR*
- *Akku*
- *Stackpointer (SP)*
- *Programcounter (PC)*

Block 2: HW- und SW-Entwicklungsumgebung

1. Welche Haupt-Funktionsgruppen des MC MiniCar kennen Sie?

   HCS08 Microcontroller

   System management (Akku, USB-Ports, OSBDM)

   Communication (Bluetooth, Infrared)

   Sensor / Actuator (LED, Switch, Button, Line Sensor, Buzzer, I2C)

   Drive (H-Bridge, Motor, optical Encoder)

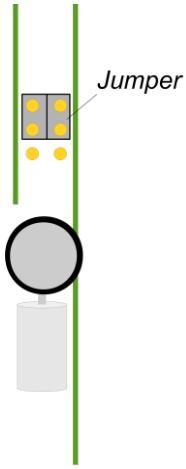
2. Wie kann man ein neues Image auf's MC-Car laden?

   Die Jumper müssen richtig gesetzt sein.

   MC-Car via USB (linker Port) mit dem PC Verbinden. (LED beim USB-Port muss leuchten)

   MC-Car einschalten. (LED zwischen den Platinen muss Rot & Grün leuchten)

   Im CodeWarrior entsprechendes Projekt auswählen und via Button in der Menu Leiste übertragen, z.B. mit „Debug LC for Simple Flash".

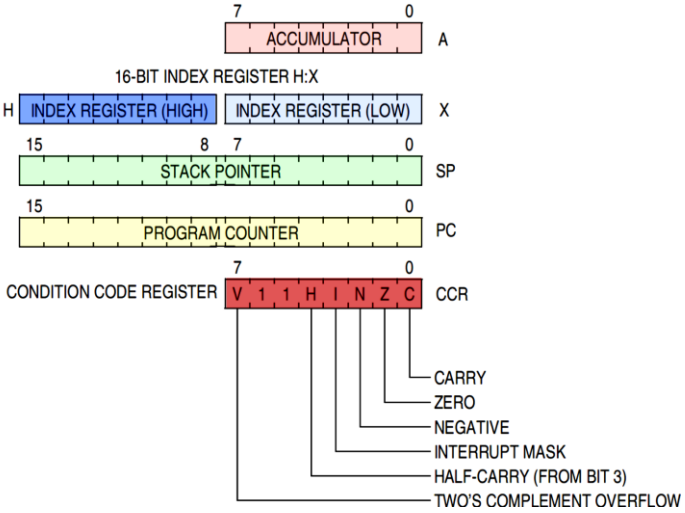


Block 3: Assembler & Einführung HCS08

1. Unter welchen Umständen würden Sie eine MCU in Assembler programmieren?
   Wenn die Rechenleistung, Speicherplatz (RAM) begrenzt ist. Wenn eine effiziente, HW-nahe Programmierung benötigt wird.

2. Welche Register in der HCS08-CPU kennen Sie?

A	*Akku (Accumulator)*	*8-Bit*
X	*H:X Index Register*	*16-Bit*
SP	*Stack Pointer*	*16-Bit*
PC	*Program Counter*	*16-Bit*
CCR	*Condition Code Register*	*8-Bit*



3. Welche Register gibt es im Mikrosim-Simulator aber nicht in der HCS08-CPU?
   OP (Operand) und INST (Instruction)-Register

4. Unter welcher Adresse kann man das Data Direction Register von Port D ansprechen?
   PTBDD 0x0003 Port B Data Direction Register
   → 0x0003

5. Welche Vor- und Nachteile hat die Befehlskombination LDA/STA gegenüber dem MOV Befehl?
   LDA/STA braucht mehr Buszyklen, dafür ist danach der Wert von Source noch immer im Akku.
   Bei 8-Bit: 3+3 = 6 Buszyklen
   Bei 16-Bit: 4+4 = 8 Buszyklen
   MOV: Akku wird nicht beeinflusst, braucht weniger Buszyklen. 16-Bit Adressen sind nicht möglich.
   Bei 8-Bit: 5 Buszyklen
   Bei 16-Bit: nicht möglich.

Block 4: Assemblerdirektiven & Adressierungsarten

1. Was bewirkt die Assembler-Direktive DS.W 3 ?

   Reserviert Speicher (RAM) für Variablen. Hier 3 Word-Variablen.

2. Wie viele Buszyklen sind für die Ausführung des Befehls LDA $10A1 erforderlich?

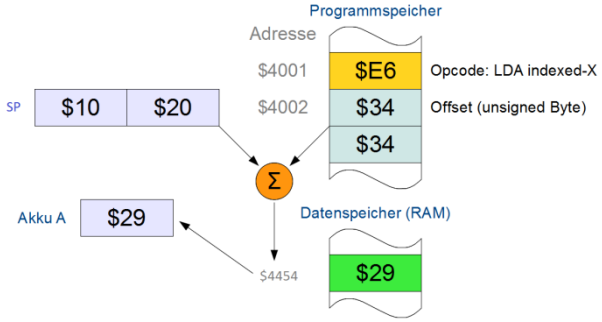
Source Form	Addr. Mode	Machine Code		HCS08 Cycles	Access Detail
		Opcode	Operand(s)		
LDA *#opr8i*	IMM	A6	ii	2	pp
LDA *opr8a*	DIR	B6	dd	3	rpp
LDA *opr16a*	EXT	C6	hh ll	4	prpp
LDA *oprx16*,X	IX2	D6	ee ff	4	prpp
LDA *oprx8*,X	IX1	E6	ff	3	rpp
LDA ,X	IX	F6		3	rfp
LDA *oprx16*,SP	SP2	9ED6	ee ff	5	pprpp
LDA *oprx8*,SP	SP1	9EE6	ff	4	prpp

   LDA opr16a → 4 Buszyklen

3. Warum gibt es keinen STA-Befehl mit immediate Adressierung?

   STA speichert den Akku im Memory, dazu muss zwingend eine Adresse angegeben werden.

4. Erklären Sie im Detail die Funktionsweise der indexierten Adressierungsart SP2 !



   Für die indexierte Adressierung wird der Stackpointer (SP) verwendet – und 16-bit Offset.

   Bsp: LDA oprx16,SP

   Nach dem OPCode wird der 16-bit Offset gelesen und zum Wert im Stackpointer (SP) addiert. Danach wird der Wert an der berechneten Adresse in den Akku geladen.

5. Wie viele Bytes Programmspeicher benötigt der Befehl LDHX $5B,SP ?

Source Form	Addr. Mode	Machine Code		HCS08 Cycles	Access Detail
		Opcode	Operand(s)		
LDHX *#opr16i*	IMM	45	jj kk	3	ppp
LDHX *opr8a*	DIR	55	dd	4	rrpp
LDHX *opr16a*	EXT	32	hh ll	5	prrpp
LDHX ,X	IX	9EAE		5	prrfp
LDHX *oprx16*,X	IX2	9EBE	ee ff	6	pprrpp
LDHX *oprx8*,X	IX1	9ECE	ff	5	prrpp
LDHX *oprx8*,SP	SP1	9EFE	ff	5	prrpp

   LDHX oprx8, SP → Opcode: OEFE ff → 6 Bytes

6. Erklären Sie den Sprungbereich der relativen Adressierung!

   Der Sprungbereich bei der relativen Adressierung beträgt -126 … + 129.

   Das auf den Opcode folgende Byte wirkt als 2er-Komplement Offset zum bereits weitergezählten Programm Counter. 8Byte wäre -128 .. + 127, da nun aber der PC +2 ist, ist der Sprungbereich -126 … + 129 relativ zum Ort wo der eigentliche Opcode für den Branch-Befehl steht.

Block 5: Adressierungsarten & Programmiertechniken

1. Welche Arten von Transport-Befehlen kennen Sie?

   LDA Load Accumulator from Memory
   LDHX Load Index Register from Memory
   LDX Load X (Index Register Low) from Memory
   MOV Move
   STA Store Accumulator in Memory
   STHX Store Index Register
   STX Store X (Index Register Low) in Memory
   TAP Transfer Accumulator to Processor Status Byte
   TAX Transfer Accumulator to X (Index Register Low)
   TPA Transfer Processor Status Byte to Accumulator
   TSX Transfer Stack Pointer to Index Register
   TXA Transfer X (Index Register Low) to Accumulator
   TXS Transfer Index Register to Stack Pointer

2. Was bewirkt der Befehl TAP bei der HCS08 CPU?

   TPA Transfer Processor Status Byte to Accumulator

   Der Inhalt vom Akku wird in das CCR (Condition Code Register), „Flag-Register“ übertragen.

3. Welche arithmetischen Flags (Condition Codes) der HCS08 CPU kennen Sie?

   C Carry
   Z Zero
   N Negative
   I Interrupt Mask
   H Half-Carry (From Bit 3)
   V Two’s Complement Overflow

4. In welchem Register wird beim HCS08 das Resultat arithmetischer Operationen gespeichert?

   Im Akku.

5. Woran erkennt die ALU ob bei den Befehlen ADD und SUB signed oder unsigned gerechnet werden muss?

   Gar nicht, es spielt für ADD / SUB keine Rolle.

Block 6: Weitere HCS08-Befehle & Assemblerprogrammierung

1. Was ist der Unterschied zwischen COMA und EOR #$FF ?

   COMA: COM-Akku: Einer Komplement des Akku (A = $FF –A)

   EOR #$FF: Bitweise Xor der einzelnen Bits. (A = A xor $FF)

2. Wie viele Zyklen erfordert die Ausführung der Befehle BCLR bzw. BSET und warum?

   5

   Zuerst muss das ganze 8-bit Register gelesen werden. Danach das Bit setzen, resp. löschen und dann wieder das ganze Register zurück speichern.

3. Welche Flag-Kombination testet der Befehl BGE und warum?

   BGE = Branch if Greater Than or Equal To

   N-Flag: Negative

   V-Flag: Two's complement overflow

   if N ⊕ V = 0 (Signed) → N (XOR) V = 0

4. Warum gibt es den Befehl BRN im Befehlssatz der HCS08-CPU?

   BRN = Branch never

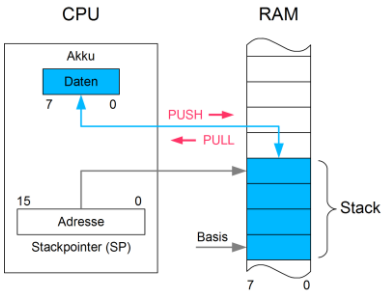
   Damit für Debug einen bestehenden BR auf BRN anpassen kann. Somit werden die Adresse des Programms nicht verändert und das Debugen ist einfacher.*

Block 7: Unterprogramme & Stack

1. Nach welchem Prinzip funktioniert ein Stack-Speicher?

   Der Stack ist ein spezieller Datenspeicher der nach dem Last-In-First-Out (LIFO) Prinzip arbeitet.

   Die Adressierung erfolgt über das Stackpointer-Register (SP) der CPU.



2. Wie wird im CW-Projekt der Stack initialisiert?

```
Stacksize:    EQU    $40
              ...
DATA:         SECTION
TofStack:     DS     Stacksize-1        ; Stack reservieren
BofStack:     DS     1
              ...
PROGRAM:      SECTION
              LDHX   #(BofStack+1)      ; Stackpointer initialisieren
              TXS                       ; SP := HX - 1
              ...
              PSHA                      ; CPU-Status retten (Akku und
              PSHX                      ; X-Register)
              ...
              PULX                      ; CPU-Status wiederherstellen
              PULA                      ; Reihenfolge beachten (LIFO!)
                                        ; Stackpointer zeigt wieder auf BofStack
```

3. Welche Informationen werden auf dem Stack gespeichert?

   Rücksprungadressen, Variablen (Parameter)

4. Welche Informationen können nicht auf dem Stack gespeichert werden?

   Globale Variablen (Konstanten) (Variablen ausserhalb einer C-Funktion werden auf dem HEAP angelegt.)

5. Nennen Sie Vor- und Nachteile von Unterprogrammen.

   Vorteile

   - *Wiederkehrende Befehlsfolgen brauchen nur einmal im Speicher abgelegt werden.*
   - *Wiederkehrende Befehlsfolgen werden nur einmal programmiert und getestet.*
   - *Programme können modular aufgebaut werden.*
   - *Programme können von mehreren Personen parallel entwickelt werden.*
   - *Teilprogramme können unabhängig voneinander übersetzt werden.*

   Nachteile

   - *Der Aufruf des Unterprogramms, Parameterübergabe und Rücksprung brauchen Zeit.*

6. Welche Arten der Parameterübergabe kennen Sie und wie werden diese beim HCS08 realisiert?

   call-by-value via Register

   call-by-reference via Stack

Block 8: Interruptsystem & Timer (Teil 1)

1. Wozu werden Interrupts benötigt?

   Auf bestimmte Ereignisse aus der internen und externen Peripherie muss ein MC-System sofort reagieren (z.B. Messwert-Überwachung, serielle Kommunikation). Der Zeitpunkt des Auftretens dieser Ereignisse ist nicht vorhersehbar.

2. Welche Vor- und Nachteile haben Polling- bzw. Interruptbetrieb?

   Interrupt = Ausnahmebehandlung

   + *Sehr kurze Reaktionszeit durch automatisches Reagieren auf Ereignis und Unterbrechung des Programms zur Abarbeitung einer Interrupt-Service-Routine (ISR)*

     → Echtzeit-fähige Systeme je nach Interrupt-Latenz möglich

   -- *Aufwendige Status-Rettung nötig, da Zeitpunkt der Programmunterbrechung unbekannt*

   Polling = Zyklisches Abfragen

   + *Kürzere Programmunterbrechung, da der Zeitpunkt der Unterbrechung bei Programmierung bekannt ist, so dass eine effizientere Status-Rettung möglich ist*

   -- *Vergeudung von Rechenzeit bei sehr seltenen Ereignissen*

3. Was ist eine ISR?

   Interrupt-Service-Routine

4. Worin unterscheiden sich ISR und Subroutine?

   ISR: Sichert die Register / den Akku und lädt diese zurück nach der Abarbeitung.

   Subroutine: Der Programmierer muss sich selber darum kümmern.

5. Was sind Interruptvektoren und wie funktionieren sie?

   >>> Project Settings > Linker Files > Project.prm

   Im Project.prm wird definiert welche Funktion bei einem Interrupt aufgerufen werden.

6. Wie viele Bits umfasst ein Counter im Timersystem des MC9S08JM60?

   16-Bit (Maximaler Wert: 2^{16} -1 = 65‘535)

7. Wie lassen sich diese Counter aus der SW auf einen bestimmten Wert setzen?

   Gar nicht (via Modulo arbeiten) (Er lässt sich nur auf 0 setzen)

8. Wie muss man vorgehen, um stets konsistente Counter-Werte auszulesen?

   Nicht speziell, jedoch müssen die High & Low-Werte ausgelesen werden, damit die MCU den Lock wieder freigibt.

9. Warum realisiert man Zeitverzögerungen mit dem Timersystem und nicht mit NOP-Befehlen?

   NOP brauch Rechenzeit (Busy-Waiting). MHz kann je nach CPU Unterschiedlich sein: ungenau.

   Mit dem Timersystem können die Zeiten genau definiert werden. (Sofern der Interrupt zeitnah abgearbeitet wird.)

Block 9/10: Output-Compare & Input-Capture

1. Welchen generellen Anwendungsfällen entsprechen die beiden Timer-Betriebsarten Output-Compare (OC) und Input-Capture (IC)?

   Output-Compare (OC): *Wecker*

   Input-Capture (IC): *Stoppuhr*

2. Erklären Sie die Funktion des Value-Registers in den Timer-Betriebsarten OC und IC.

   Output-Compare (OC): *Value Register um den Comparator zu setzen*

   Input-Capture (IC): *Register wird von TPM gesetzt: TimeTicks zwischen den Interrupts*

3. Warum ist es vorteilhaft das Timer-Modul mit einem Modulo-Wert von 0xFFFF zu betreiben?

   Alle weitere Channels sind davon abhängig.

   Für einen „Counter“ kann einfach dazu addiert werden, bei 0xFFFF wird einfach wieder bei 0 begonnen.

4. Wie lässt sich mit dem Timer ein Output-Compare Interrupt generieren ohne den Wert am Port-Pin zu ändern?

   Die Aktion von OC deaktivieren (ELS Bit entsprechend setzen).

5. Wie heissen die zwei Betriebsmodi eines Logic Analyzers und worin unterscheiden sie sich?

   Zustands-Analyse (synchron)

   Timing-Analyse (asynchron)

Block 11: Pulse-Width Modulation (PWM)

1. Wie wird in der Timer-Betriebsart Edge-Aligned PWM Periodendauer und Duty Cycle des generierten Signals bestimmt?
   Periodendauer: MOD (TPM2MOD)
   Duty Cycle: Channel Value (TPM2C0V)

2. Warum bietet es sich beim MC-Car an, die PWM-Signale für linken und rechten Motor mit TPM2 und nicht mit TPM1 zu generieren?
   Damit wir unabhängig der restlichen Timer, resp. Channel den MOD Wert des Timers setzten können.
   Anzahl Kanäle: TPM1 hat 6 Channels, TPM2 hat 2 Channels.

3. Was muss man für die Generierung eines Signals mit 100% Duty Cycle beachten?
   Die Channel Value (TPM2C0V) muss zwingend grösser als MOD (TPM2MOD) sein! (Mind. 1 grösser)

4. Was ist Vor- und Nachteil der Betriebsart Center-Aligned PWM?
   Bei Edge-Aligned werden die Channels immer zur gleichen Zeit aktiviert. Dies kann zu elektromagnetische Verträglichkeit (EMV) - Störungen führen.
   Bei Center-Aligned ist dies nicht / weniger der Fall.
   Bei Center-Aligned werden zwei Channels dafür verwendet!
   Natürlich werden dadurch die Timers / Channels zu unterschiedlichen Zeiten aus-/eingeschalten – dies ist z.B. für Schritt-Motoren ungeeignet.

Block 12/13: IIC-Bus

1. Wie viele Leitungen benötigt der IIC-Bus?
   2 bidirektionale Leitungen (Clock SCL, Daten SDA)

2. Mit welcher Bitrate wird beim IIC-Bus gearbeitet?
   *Standard 100 kHz; **Fast 400 kHz**; Fast Plus 1 MHz; High Speed 3,4 MHz*

3. Welche Treiberstufen benötigen die IIC-Bus Teilnehmer und warum?
   Alle Busteilnehmer benötigen Open-Drain Ausgangstufen (kein aktiver H-Pegel möglich).
   Externe Pullup-Widerstände sorgen für H-Pegel (Default-Zustand).

4. Wie werden Start und Stop einer Übertragung signalisiert?
   *Start-/Stop-Bedingungen werden immer vom **Master** generiert und können durch andere Master und Slaves als Protokollverletzung von normalen Datenbits unterschieden werden.*
   Nach einer Start-Bedingung ist der Bus busy, nach einer Stop-Bedingung wieder idle.

5. Wann dürfen Daten auf der SDA-Leitung geändert werden?
   SDA wird geändert während SCL = 0 (aktiv) ist, und ausgewertet wenn SCL = 1 (inaktiv) ist.

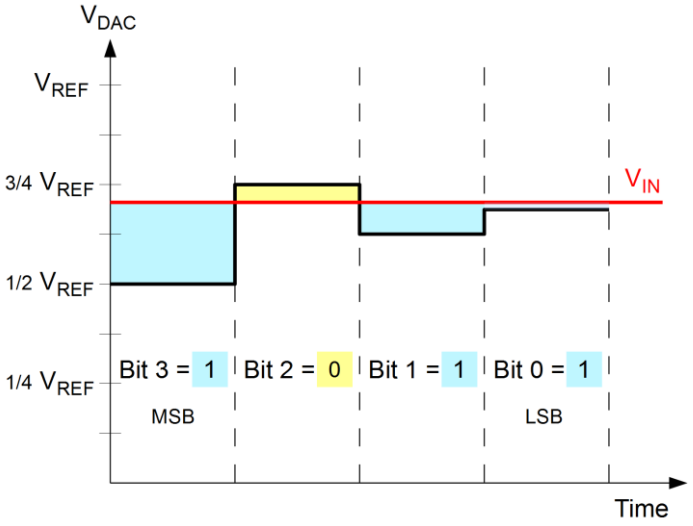
6. Wie entscheidet sich wer beim IIC-Bus Master und Slave ist?
   Via Control Register kann definiert werden ob der Master oder Slave Modus aktiv ist. Viele Sensoren sind per Default immer Slave (Hardware).
   Slave kann nur Antworten auf Anfragen von Master.

7. Wie entscheidet sich wer beim IIC-Bus Sender und Empfänger ist?
   Ob ein Busteilnehmer (irrelevant ob Master oder Slave) Daten senden oder empfangen kann, entscheidet das Read/Write Bit (R/W). Das R/W Bit ist das Bit0 der ersten 8 übertragenen Bits. (Bit 1-7 werden als Teilnehmer-Adresse gebraucht).
   Ist R/W=0, so sendet der Master zum Slave.
   Ist R/W=1, so empfängt der Master vom Slave.

8. Wozu wird beim IIC-Bus die "Repeated-Start"Bedingung verwendet?
   Eine Repeated-Start (Sr) Bedingung kann vom aktiven Master an Stelle einer Stop-Bedingungen generiert werden, wenn er den Bus weiter belegen will.

Block 14: A/D-Wandler

1. Erklären Sie das Prinzip der A/D-Wandlung mittels sukzessiver Approximation.
   Er vergleicht das vorhandene Signal mit einem Signal das generiert wird. Je nachdem wird um die Hälfte erhöht oder verringert.



- *Unter Verwendung eines im A/D-Wandler integrierten D/A-Wandlers, werden nacheinander verschiedene Vergleichsspannungen gebildet.*
- *Beim MSB beginnend, nähert sich so der Wert des generierten Digitalwortes schrittweise der Eingangsspannung an.*
- *Je mehr Bits das generierte Digitalwort umfasst, um so genauer wird die Näherung sein.*
- *Während der Wandlung wird die Eingangsspannung konstant gehalten (Sample & Hold).*

2. Wie viele Kanäle besitzt der A/D-Wandler im MC9S08JM60?
   12

3. Wie viele Anologsignale kann der A/D-Wandler im MC9S08JM60 gleichzeitig wandeln?
   1

4. Mit welcher Bit-Auflösung können Analogsignale im MC9S08JM60 eingelesen werden?
   8, 10 und 12Bit.

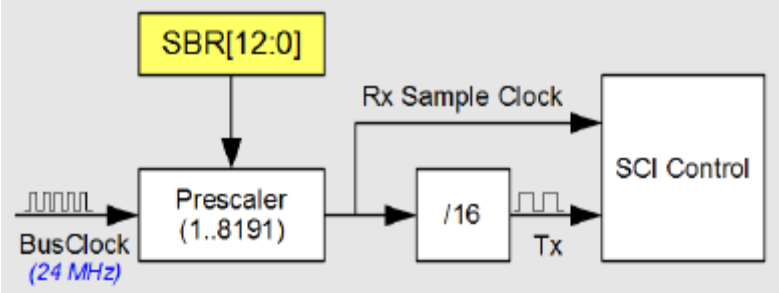
Block 15/16: RS232 Serielle Kommunikation

1. Wie viele Leitungen benötigt ein RS-232-Interface zur Datenübertragung?
   2 (Rx und Tx Datenleitung)

2. Warum gibt es bei RS-232 im Gegensatz zu IIC keine Clock-Leitung?
   Das Timing ist asynchron, d.h. Sender und Empfänger verwenden einen eigenen Taktgeber. Damit die Datenübertragung funktioniert müssen beide Devices auf die Gleiche Baudrate eingestellt sein.

3. Wie viele Datebytes kann man mit 9600 Baud/s und einem RS-232-Format von 8-No-1 pro Sekunde übertragen?
   9600 Baud, 8 Daten Bits, Odd Parity, 1 Stop Bit
   → Pro 8 Data Bits + 1 Start Bit, 1 Parity Bit, 1 Stop Bit = 11 Bits
   9600 / 11 = 872 Bits → 109 Bytes können übertragen werden pro Sekunde

4. Wie erklärt sich der Faktor 16 bei der Berechnung des BR-Wertes für das SCI-Modul im MC9S08JM60?
   BR = BaudRate
   Der Takt wird immer durch 16 dividiert vor dem SCI Control:



$$Baud\ Rate = \frac{f_{BusClock}}{16 \cdot SBR}$$

Block 17/18: RTOS

1. Wie wird die Wartezeit implementiert?

   Mit einem Delay:

   ```
   OSTimeDly(50); // 50 * 20ms = 1'000ms delay
   ```

2. Was müssen Sie beachten, wenn Sie zwei LEDs auf dem gleichen Port blinken lassen wollen?

   Nur jeweils das entsprechende Bit ändern (und nicht der ganze Port, alle Bits des Ports).

- ⇨ Echtzeit-OS → Garantierte Reaktionszeit, d.h. der Prozess mit der höchsten Priorität wird garantiert zur gewünschten Zeit ausgeführt.
- ⇨ Bei uns wird alle 20ms der Scheduler aufgerufen, d.h. alle 20ms kann es ein Kontext-Switch geben!

- ⇨ Wenn kein Task läuft, läuft der idle-Task (dieser braucht auch Stack!)
- ⇨ Wenn ein Interrupt aufgerufen wird, wird der Stack vom aktuell laufenden Task verwendet – im Fall vom idle Task kann der Stack sehr klein sein.