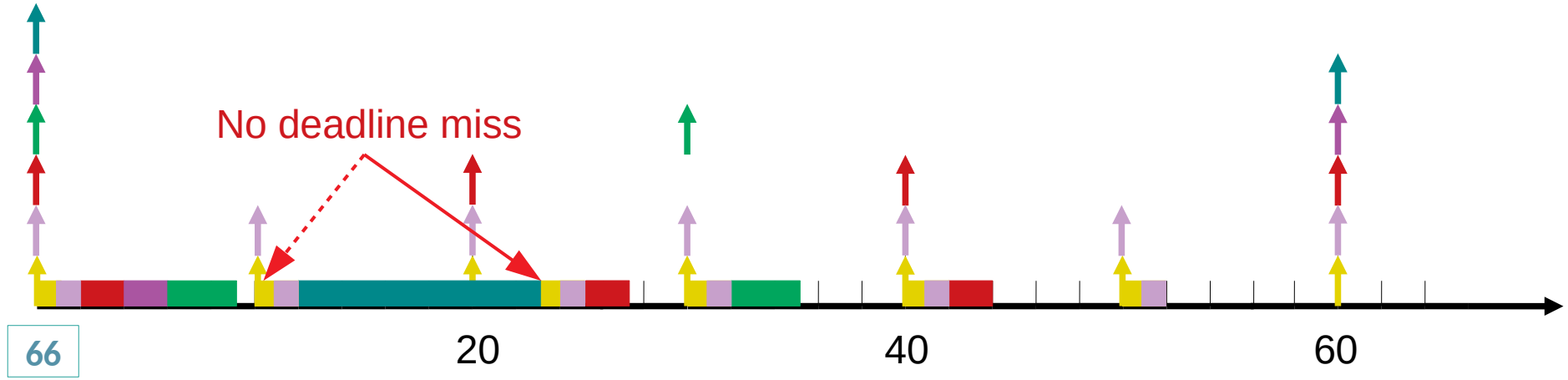
Static (off-line) scheduling

Scheduling table: implementation option 1

- Implementing the scheduling table **with no preemption**
  - Cyclic executive whose period is the hyperperiod of the tasks
  - Tasks with smaller periods are called multiple times (multiple jobs) and their calls are **synchronized in time**
    - e.g. Reading a sensor, and making sure each read gets a fresh value
  - Synchronization is ensured using hardware features
    - Busy waiting while polling a Time Stamp Counter register
    - Polling from a timer



Static (off-line) scheduling

Scheduling table

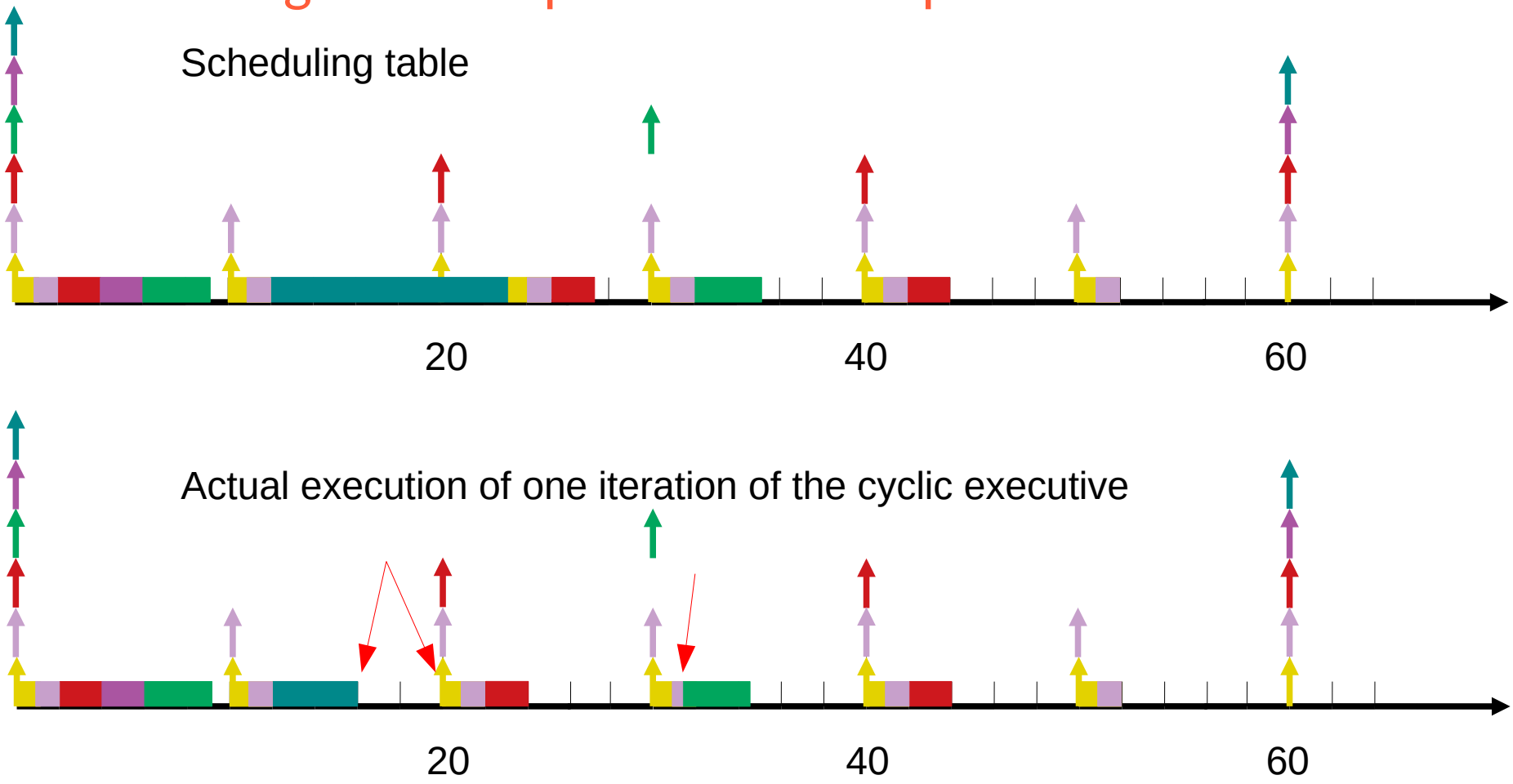
Synchronize according to data arrival (harmonic tasks)

`wait_timer()` is based on a timer which counts periods of 10 ms.

```
loop :
    wait_timer();
    pedal_angle();
    speed();
    engine_rotation();
    collision_detection();
    ECU();
    wait_timer();//10ms
    pedal_angle();
    speed();
    airbag();
    wait_timer();//10ms
    pedal_angle();
    speed();
    engine_rotation();
    wait_timer();//10ms
    ...
```

Static (off-line) scheduling

Scheduling table: implementation option 1



Static (off-line) scheduling

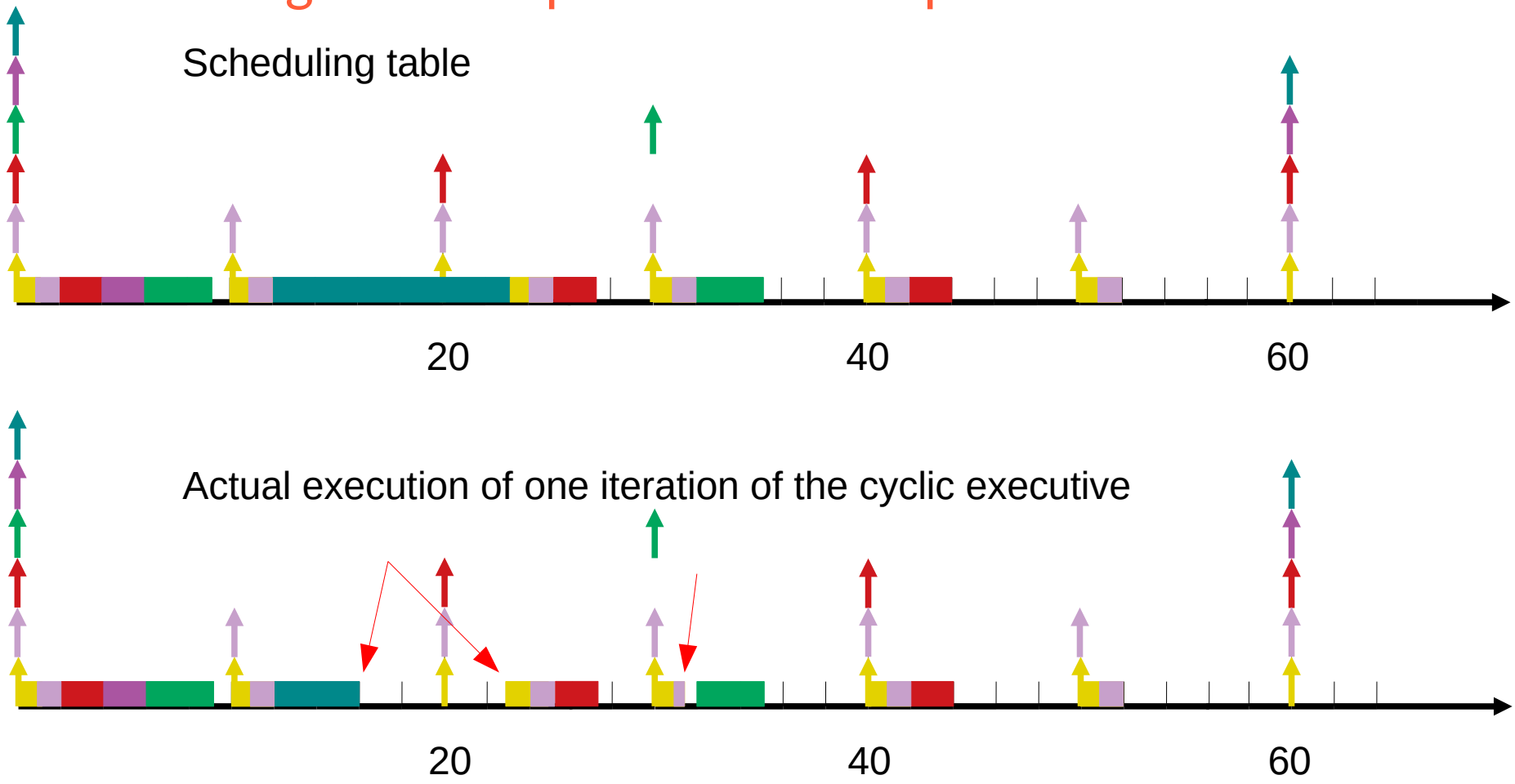
Scheduling table

Synchronize according to starting dates in the table

```
loop :
    wait_timer();
    pedal_angle();
    wait_timer();//1ms
    speed();
    wait_timer();//1ms
    engine_rotation();
    wait_timer();//2ms
    collision_detection();
    wait_timer();//2ms
    ECU();
    wait_timer();//3ms
    pedal_angle();
    wait_timer();//1ms
    speed();
    wait_timer();//1ms
    airbag();
    wait_timer();//12ms
    ...
```

Static (off-line) scheduling

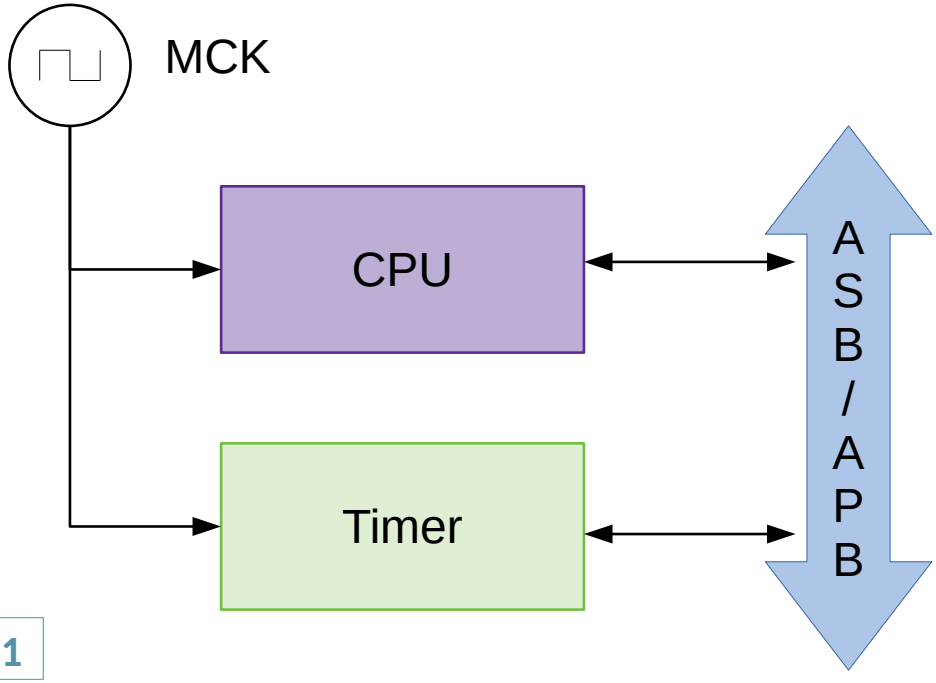
Scheduling table: implementation option 1

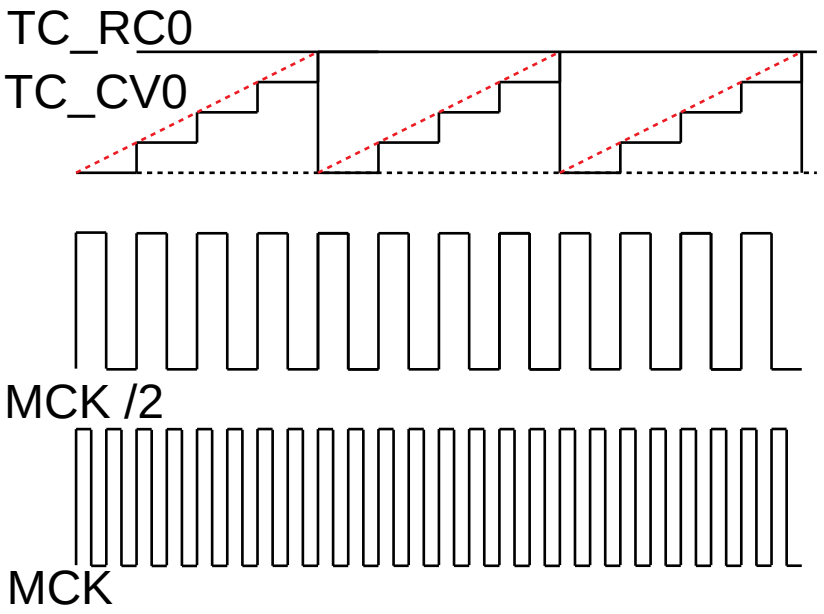


Static (off-line) scheduling

Timer:

- Hardware counter (16-bits on AT91SAM7S which we will simulate)
- Incremented at a rate depending on the master clock
  - Master clock is ~48MHZ on the AT91
  - The counter can divide the frequency to count for longer periods
  - Programmed using the CPU through the system and peripheral buses

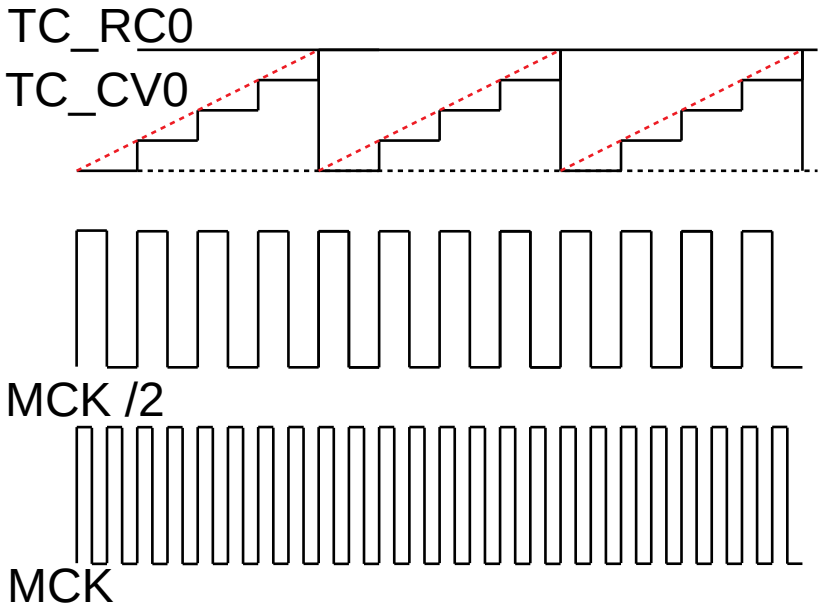




Static (off-line) scheduling

Timer:

- CCR – command register
  - CLKDIS : disable bit
  - CLKEN : enable bit
  - SWTRG : reset bit
- RC – 16 bits register (contains the threshold)
- CV – Counter value (incremented each MCK/DIV tick)
- CMR – configure
  - DIV : MCK divider
  - CLOCK1-5 : divider 2,8,32, 128,1024
  - CPCTRG : reset CV when CV = RC
- SR – Status register
  - CPCS : bit set when CV = RC



Static (off-line) scheduling

Timer:

- CCR – command register
  - CLKDIS : disable bit
  - CLKEN : enable bit
  - SWTRG : reset bit
- RC – 16 bits register
- CV – Counter value

- CMR – configure
  - DIV : MCK divider
  - CLOCK1-5 : divider 2,8,32, 128,1024
  - CPCTRG : reset CV when CV = RC
- SR – Status register
  - CPCS : bit set when CV = RC

Example: set the timer to count seconds

MCK : 48MHz => 48,000,000 ticks per second

RC : 16 bits register => can count to 65,536
If we want the timer to count seconds:
48,000,000 / divider ≤ 65,536
<=> divider ≥ 48,000,000/65,536
<=> divider ≥ 732.4
=> must select CLOCK5 (divide by 1024)

Then: 48,000,000/1024 ticks of TC <=> 1s
=> 46,875 ticks of TC <=> 1s
=> 1s is reached when CV = 46,875
=> set RC to 46,875

Static (off-line) scheduling

Timer:

- CCR – command register
  - CLKDIS : disable bit
  - CLKEN : enable bit
  - SWTRG : reset bit
- RC – 16 bits register
- CV – Counter value

```
// initialize main timer
 TC0_CCR = TC_CLKDIS; /* disable timer */
 TC0_CMR = TC_CLOCK5  /* Divide by 1024*/
           | TC_CPCTRG; /* Restart when CV = RC */
 TC0_RC = 46875; /* Delay = 1s  */
 TC0_CCR = TC_CLKEN; /* Enable TC0 */
 TC0_CCR = TC_SWTRG ;  /*  Reset TC0 value */
```

- CMR – configure
  - DIV : MCK divider
  - CLOCK1-5 : divider 2,8,32, 128,1024
  - CPCTRG : reset CV when CV = RC
- SR – Status register
  - CPCS : bit set when CV = RC

```
// wait for the end of count
 while(!(TC0_SR & TC0_CPCS));
```

Static (off-line) scheduling

Scheduling table

Case 1: harmonic periods

Synchronize according to data arrival

Timer must be able to count 10 ms (fastest period)

```
loop :
    wait_timer();
    pedal_angle();
    speed();
    engine_rotation();
    collision_detection();
    ECU();
    wait_timer();//10ms
    pedal_angle();
    speed();
    airbag();
    wait_timer();//10ms
    pedal_angle();
    speed();
    engine_rotation();
    wait_timer();//10ms
    ...
```

Static (off-line) scheduling

Timer:

- CCR – command register
  - CLKDIS : disable bit
  - CLKEN : enable bit
  - SWTRG : reset bit
- CMR – configure
  - DIV : MCK divider
  - CLOCK1-5 : divider 2,8,32, 128,1024
  - CPCTRG : reset CV when CV = RC
- RC – Register
- SR – Status register
  - CPCS : bit set when CV = RC

Case 1: we want to count 10 ms

MCK : 48MHz => 48,000,000 ticks per second
=> 48,000 ticks per millisecond

RC : 16 bits register => can count to 65536
48,000,000/divider <=> 1s
480,000/divider <=> 10ms
=> 480,000/65536 ≤ divider
=> 7.32 ≤ divider
=> divider = 8
=> #define TC_DIV TC_CLOCK2

For the timer:
48,000,000/8 <=> 1s
480,000/8 <=> 10ms
60,000 <=> 10ms
=> RC = 60000 ;

Static (off-line) scheduling

```
// initialize main timer
 TC0_CCR = TC_CLKDIS; /* disable timer */
 TC0_CMR = TC_DIV | TC_CPCTRG; /* configure timer counter and restart when CV = RC */
 TC0_RC = 10_MILLISECOND; /* delay = 10ms (10*TC_MCK / (TC_DIV*1000)) */
 TC0_CCR = TC_SWTRG | TC_CLKEN; /* Enable TC0 and reset TC0 value */
```

Synchronizing on data arrival

```
//loop
while(1){
     pedal_angle() ;
     speed() ;
     engine_rotation();
     collision_detection();
     ECU();
     while(!(TC0_SR & TC_CPCS)); /* After date = 10ms */
     pedal_angle() ;
     speed() ;
     airbag() ;
     while(!(TC0_SR & TC_CPCS));/* After date = 20ms */
     …
     while(!(TC0_SR & TC_CPCS)); /* After date = 50ms */
     pedal_angle() ;
     speed() ;
     while(!(TC0_SR & TC_CPCS)); /* After date = 60ms */
}
```

Static (off-line) scheduling

```
// initialize main timer
 TC0_CCR = TC_CLKDIS; /* disable timer */
 TC0_CMR = TC_DIV | TC_CPCTRG; /* configure timer counter and restart when CV = RC */
 TC0_RC = 10_MILLISECOND; /* delay = 10ms (10*TC_MCK / (TC_DIV*1000)) */
 TC0_CCR = TC_SWTRG | TC_CLKEN; /* Enable TC0 and reset TC0 value */
```

Synchronizing on data arrival

```
//loop
while(1){
     pedal_angle() ;
     speed() ;
     engine_rotation();
     collision_detection();
     ECU();
     while(!(TC0_SR & TC_CPCS)); /* After date = 10ms */
     pedal_angle() ;
     speed() ;
     airbag() ;
     while(!(TC0_SR & TC_CPCS));/* After date = 20ms */
     …
     while(!(TC0_SR & TC_CPCS)); /* After date = 50ms */
     pedal_angle() ;
     speed() ;
     while(!(TC0_SR & TC_CPCS)); /* After date = 60ms */
}
```

Harmonic periods => no need to reconfigure the timer

Static (off-line) scheduling

Scheduling table

Case 2: non-harmonic periods, or wish to implement scheduling table

Synchronize according to starting dates

Timer must be able to count up to 12 ms (largest duration between two function calls)

```
loop :
    wait_timer();
    pedal_angle();
    wait_timer();//1ms
    speed();
    wait_timer();//1ms
    engine_rotation();
    wait_timer();//2ms
    collision_detection();
    wait_timer();//2ms
    ECU();
    wait_timer();//3ms
    pedal_angle();
    wait_timer();//1ms
    speed();
    wait_timer();//1ms
    airbag();
    wait_timer();//12ms
    ...
```

Static (off-line) scheduling

Timer:

- CCR – command register
  - CLKDIS : disable bit
  - CLKEN : enable bit
  - SWTRG : reset bit
- CMR – configure
  - DIV : MCK divider
  - CLOCK1-5 : divider 2,8,32, 128,1024
  - CPCTRG : reset CV when CV = RC
- RC – Register
- SR – Status register
  - CPCS : bit set when CV = RC

Case 2: we want to count up to 12 ms

MCK : 48MHz => 48,000,000 ticks per second
=> 48,000 ticks per millisecond

RC : 16 bits register => can count to 65536
48,000,000/divider <=> 1s
576,000/divider <=> 12ms
=> 576,000/65536 ≤ divider
=> 8.78 ≤ divider
=> divider = 32
=> #define TC_DIV TC_CLOCK3

For the timer:
48,000,000/32 <=> 1s
576,000/32 <=> 12ms
18,000 <=> 12ms
=> RC = 18000; // for 12ms
=> RC = 1500; // for 1ms

Static (off-line) scheduling

```
// initialize main timer
 TC0_CCR = TC_CLKDIS; /* disable timer */
 TC0_CMR = TC_DIV | TC_CPCTRG; /* configure timer counter and restart when CV = RC */
 TC0_RC = 1_MILLISECOND; /* delay = 1ms (TC_MCK / (TC_DIV*1000)) */
 TC0_CCR = TC_SWTRG | TC_CLKEN; /* Enable TC0 and reset TC0 value */
```

Synchronizing on schedule

```
//Cyclic executive
while(1){
      pedal_angle() ;
      while(!(TC0_SR & TC_CPCS));  /* After date = 1ms */
      speed() ;
      while(!(TC0_SR & TC_CPCS));/* After date = 2ms */
      TC0_RC = 2_MILLISECOND;
      engine_rotation();
      while(!(TC0_SR & TC_CPCS));/* After date = 4ms */
      collision_detection();
      while(!(TC0_SR & TC_CPCS));/* After date = 6ms */
      TC0_RC = 4_MILLISECOND;
      ECU();
      while(!(TC0_SR & TC_CPCS));/* After date = 10ms */
      TC0_RC = 1_MILLISECOND;
      pedal_angle() ;
      while(!(TC0_SR & TC_CPCS));/* After date = 11ms */
      speed() ;

      …
      speed() ;
      while(!(TC0_SR & TC_CPCS));/* After date = 60ms */
}
```

Static (off-line) scheduling

```
// initialize main timer
 TC0_CCR = TC_CLKDIS; /* disable timer */
 TC0_CMR = TC_DIV | TC_CPCTRG; /* configure timer counter and restart when CV = RC */
 TC0_RC = 1_MILLISECOND; /* delay = 1ms (TC_MCK / (TC_DIV*1000)) */
 TC0_CCR = TC_SWTRG | TC_CLKEN; /* Enable TC0 and reset TC0 value */
```

Synchronizing on schedule

```
//Cyclic executive
while(1){
     pedal_angle() ;
     while(!(TC0_SR & TC_CPCS));  /* After date = 1ms */
     speed() ;
     while(!(TC0_SR & TC_CPCS));/* After date = 2ms */
     TC0_RC = 2_MILLISECOND;
     engine_rotation();
     while(!(TC0_SR & TC_CPCS));/* After date = 4ms */
     collision_detection();
     while(!(TC0_SR & TC_CPCS));/* After date = 6ms */
     TC0_RC = 4_MILLISECOND;
     ECU();
     while(!(TC0_SR & TC_CPCS));/* After date = 10ms */
     TC0_RC = 1_MILLISECOND;
     pedal_angle() ;
     while(!(TC0_SR & TC_CPCS));/* After date = 11ms */
     speed() ;

     …
     speed() ;
     while(!(TC0_SR & TC_CPCS));/* After date = 60ms */
}
```

Resetting the TC must be completed before next sync

Static (off-line) scheduling

```
// initialize main timer
 TC0_CCR = TC_CLKDIS; /* disable timer */
 TC0_CMR = TC_DIV | TC_CPCTRG; /* configure timer counter and restart when CV = RC */
 TC0_RC = 1_MILLISECOND; /* delay = 1ms (TC_MCK / (TC_DIV*1000)) */
 TC0_CCR = TC_SWTRG | TC_CLKEN; /* Enable TC0 and reset TC0 value */
```

Synchronizing on schedule

```
//Cyclic executive
while(1){
      pedal_angle() ;
      while(!(TC0_SR & TC_CPCS));  /* After date = 1ms */
      speed() ;
      while(!(TC0_SR & TC_CPCS));/* After date = 2ms */
      TC0_RC = 2_MILLISECOND;
      engine_rotation();
      while(!(TC0_SR & TC_CPCS));/* After date = 4ms */
      collision_detection();
      while(!(TC0_SR & TC_CPCS));/* After date = 6ms */
      TC0_RC = 4_MILLISECOND;
      ECU();
      while(!(TC0_SR & TC_CPCS));/* After date = 10ms */
      TC0_RC = 1_MILLISECOND;
      pedal_angle() ;
      while(!(TC0_SR & TC_CPCS));/* After date = 11ms */
      speed() ;

      …
      speed() ;
      while(!(TC0_SR & TC_CPCS));/* After date = 60ms */
}
```

Must be accounted for in WCETs

Static (off-line) scheduling

Exercise 1:

- Given the following task system, write a corresponding cyclic executive (if one exists), including the timer initial configuration and the required synchronizations
  - T0 : T0 = 10ms, C0 = 2ms
  - T1 : T1 = 50ms, C1 = 15ms
  - T2 : T2 = 100ms, C2 = 16ms

Static (off-line) scheduling

Exercise 2:

- Given the following task system, write a corresponding cyclic executive (if one exists), including the timer initial configuration and the required synchronizations
  - T0 : T0 = 20ms, C0 = 5ms
  - T1 : T1 = 30ms, C1 = 5ms
  - T2 : T2 = 60ms, C2 = 10ms

Static (off-line) scheduling

Practical work:

- Develop a cyclic executive generator **for harmonic task systems**
  - Input: textual description of a scheduling table
  - Output: .c file with function corresponding to cyclic executive (including the timer configurations and the synchronizations for our AT91 target)

- If you have time, add non-harmonic periods (in case of non-integer values for RC, print an error message instead of generating the output file)