FFT-Analyse

a presentation by Piri Daniel

28. Jan. 2008

Inhalt

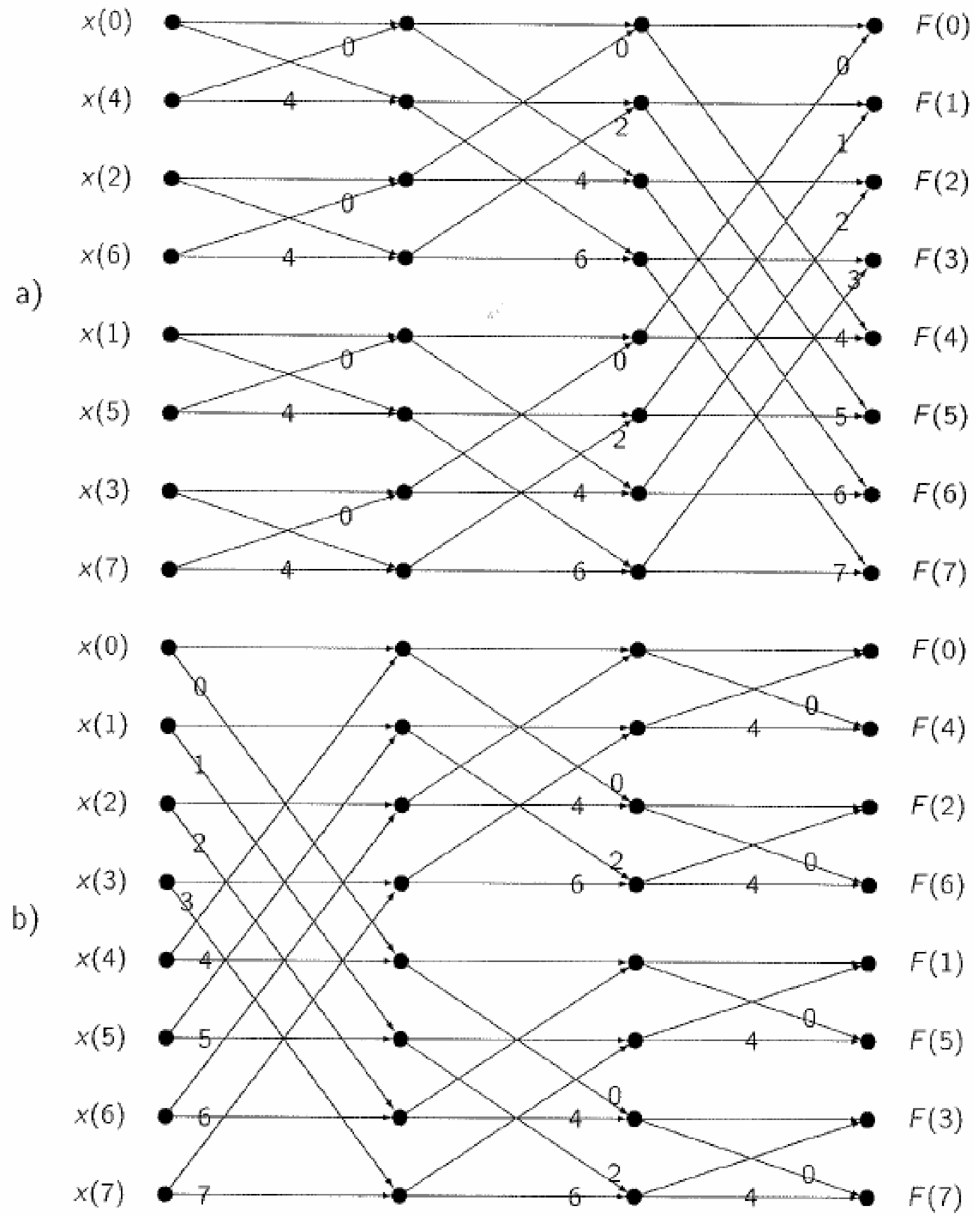
- THEORIE: Fourier Transform (diskrete und schnelle)
- PRAXIS: Anwendungen, Beispiele der FFT
- Mikrocontroller Bsp. dsPIC30F4013
- Berechnungen Excel, Simulation MathCAD

Fourier Transformation

1. Abtastung
   a. zeitkontinuierliche Signale => zeitdiskrete Signale
   b. perioden-synchrone, -asynchrone
   c. Abtastbedingungen (Shannon)
2. DFT
   a. Zeitumordnung
   b. Skalierung
   c. FFT-Zerlegung

$$\hat{x}_k = \sum_{n=0}^{N-1} x_n e^{-i2\pi \frac{kn}{N}}$$

$$x_n = \frac{1}{N} \sum_{k=0}^{N-1} \hat{x}_k e^{i2\pi \frac{kn}{N}}$$



d. Bit reversing

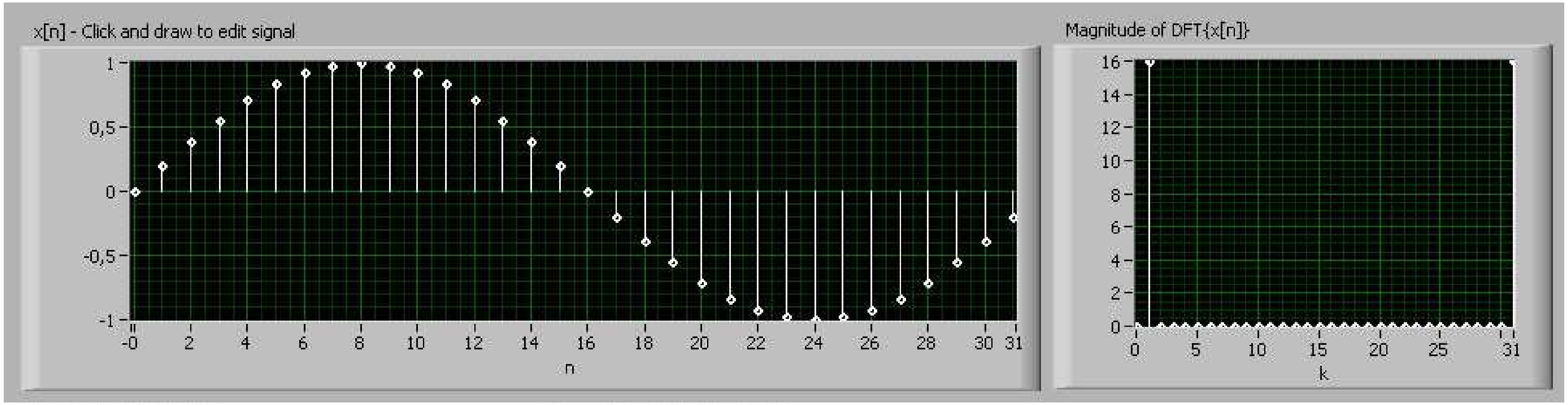
3. Z-Berechnen aus Real- und Imaginärteil

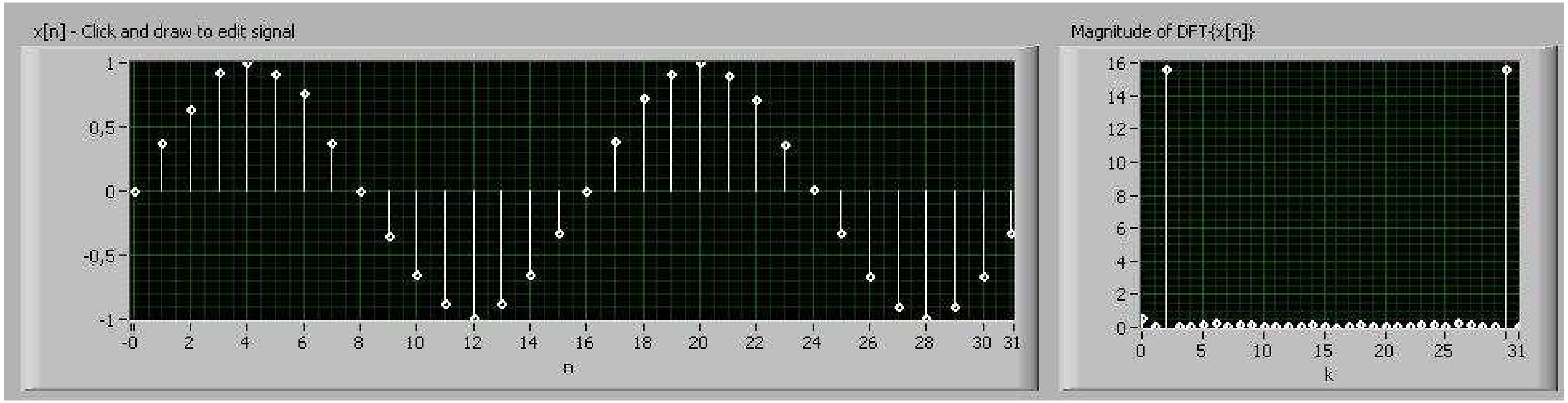
4. Auswertung

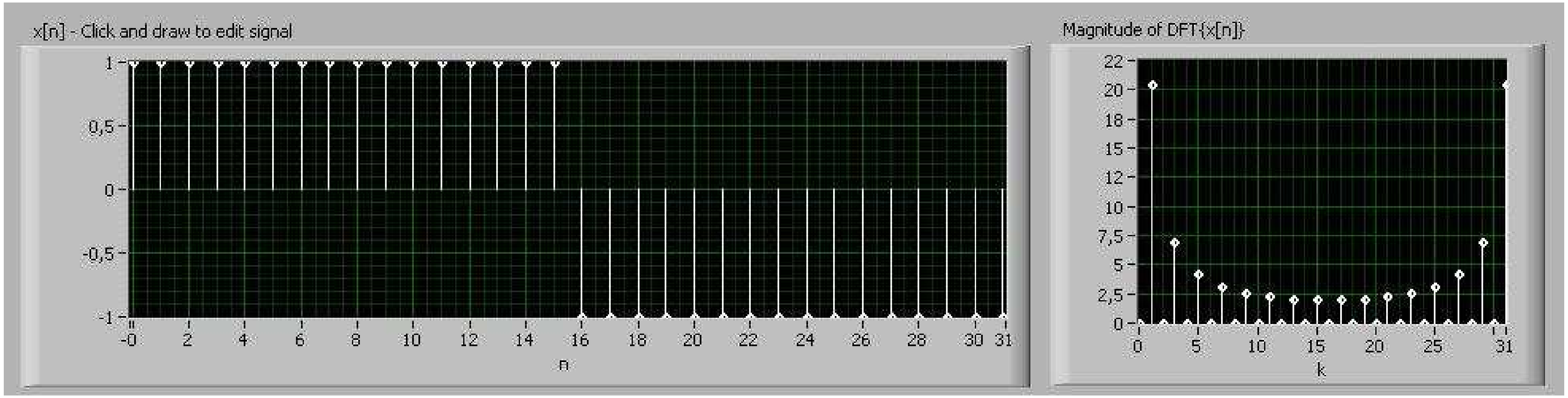
a. Frequenz und Intensität des größten Spektrums

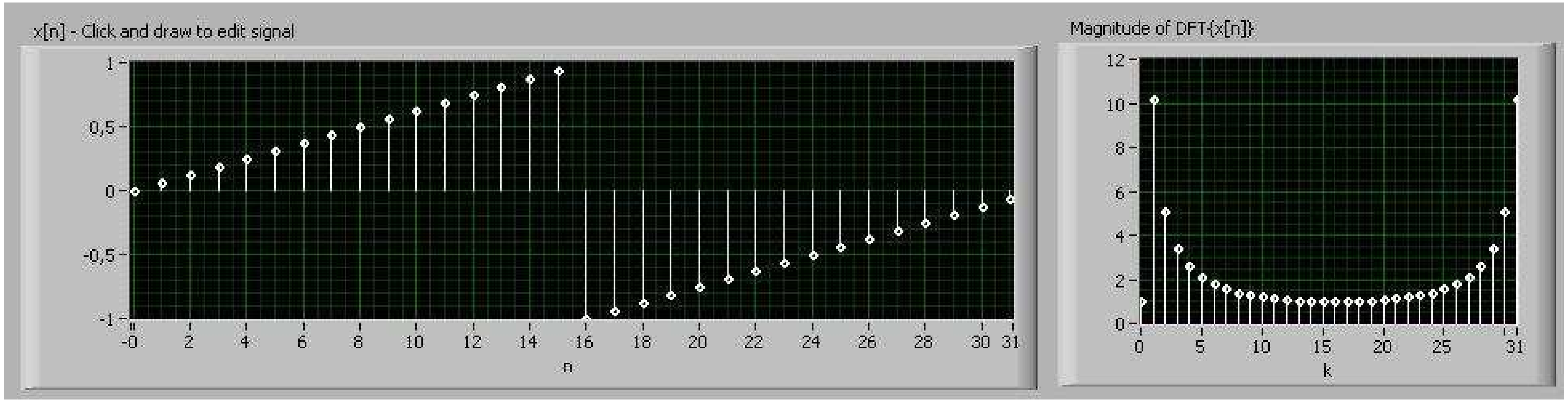
b. Rückschlüsse auf Signalform, Harmonische, etc

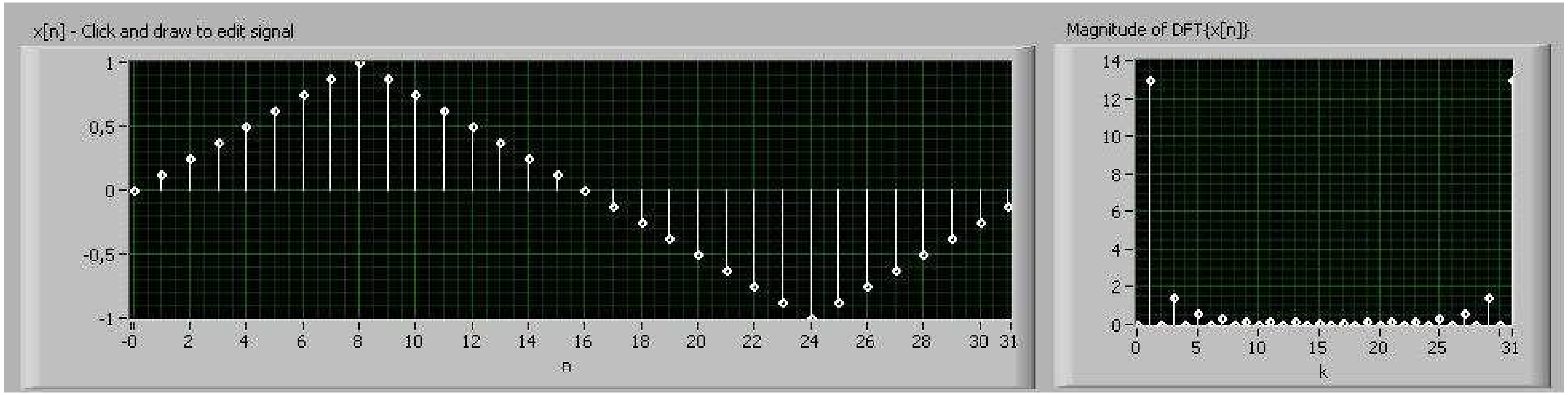
DFT typischer periodischer Signale



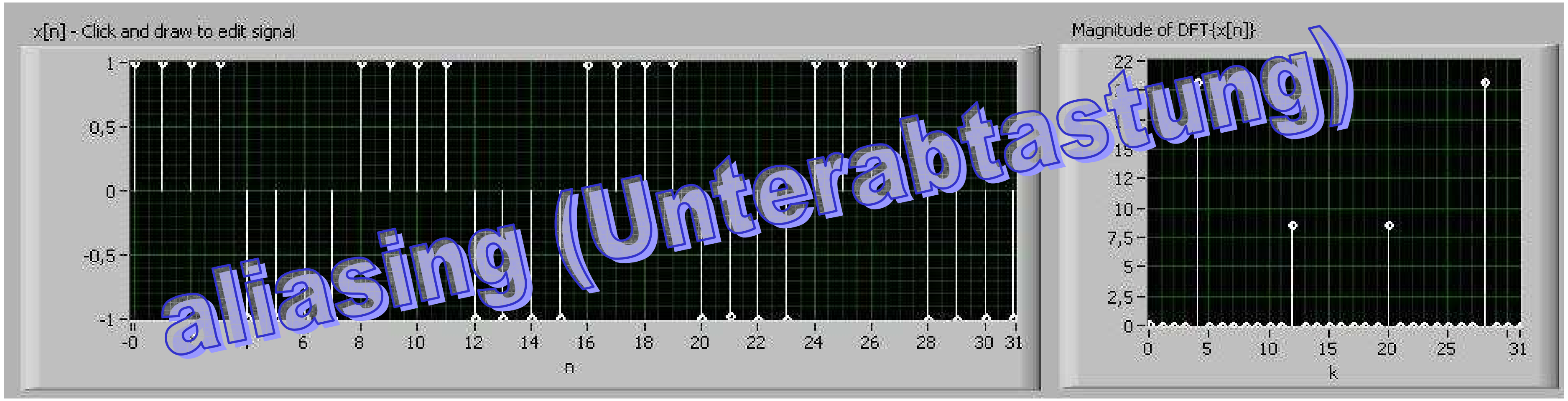



x[n] - Click and draw to edit signal
n
Magnitude of DFT{x[n]}
k


x[n] - Click and draw to edit signal
n
Magnitude of DFT{x[n]}
k



So sollte es _nicht_ ausschauen:



Praxis

Man muss unterscheiden, ob

- ein periodisches oder n.p. Signal vorliegt,
- ob die Abtastung mehrere Perioden umfasst, oder mittels HAMMING-Fenster nur Eine (Idealfall),
- und ob man periodisch Blöcke abarbeitet, oder kontinuierlich workt.

Man muss festlegen:

- Auflösung (=Samplewerte, bei FFT 2^n)
- Samplezeit (min. 2 sampl / Tsignal)

Rekonstruktion

- äquidistante zeitdiskrete Signale => zeitkontinuierliche Signale
- ursprüngliches Signal einfach nachbildbar
- durch LAGRAN-Interpolation

$$x(t) = \sum_{k=1}^{K} x(t_k) \frac{P_K(t)}{(t - t_k) P'_K(t_k)}$$

dsPIC30F4013

- 16 bit digital signal controller

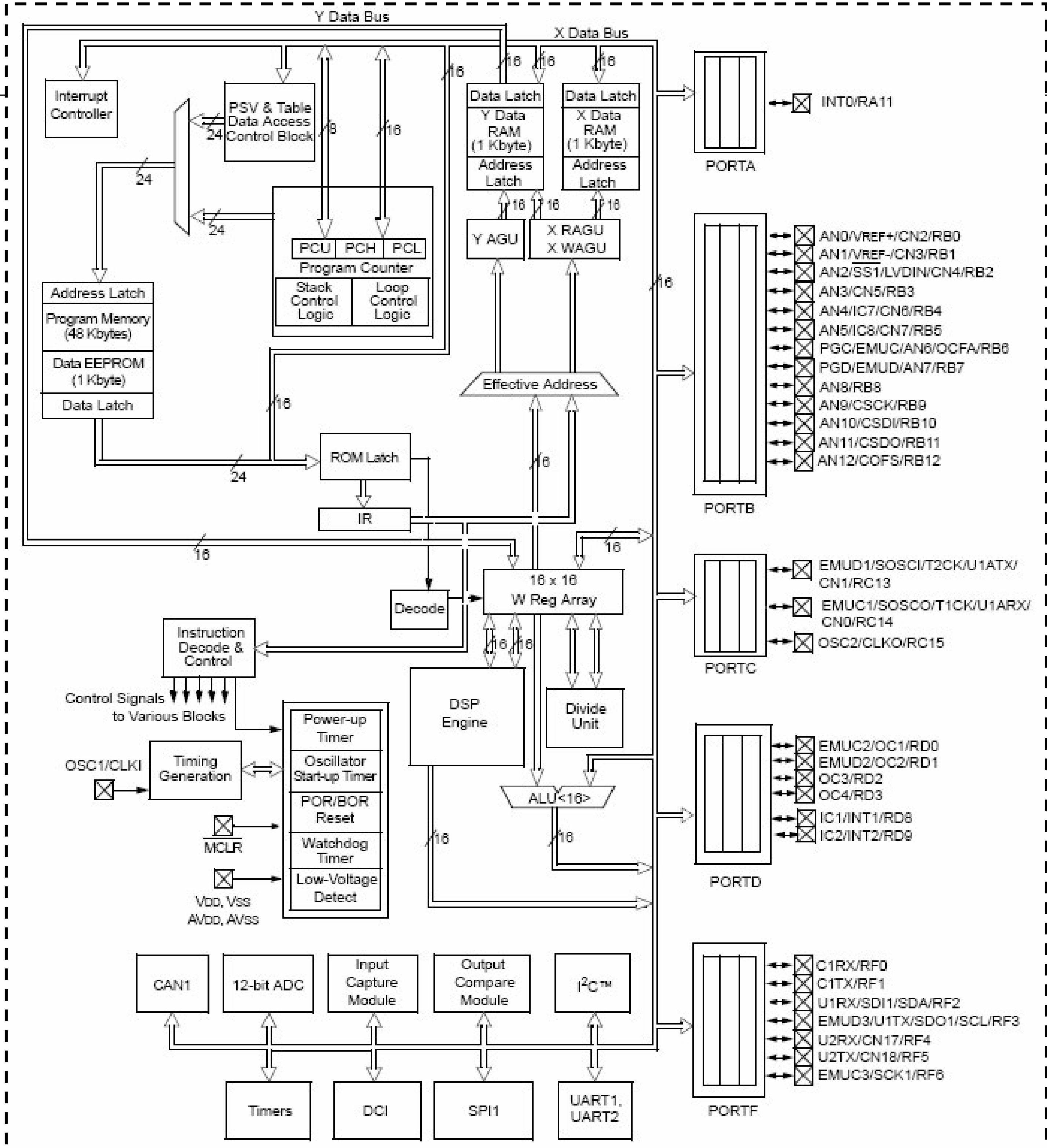
Parameter Name	Value
Vdd Range	2.5 to 5.5
UART	2
SPI™	1
SRAM Bytes	2,048
Program (FLASH) KBytes	48
Pin Count	40
Output Compare/Std. PWM	4
Input Capture	4
ICSP	Yes
I^2C™ Compatible	1
CPU Speed in MIPs	30
Codec	Yes
CAN	1
Oscillator	7.37 MHz, 512 kHz
ADC Bit	12
Watch Dog Timer	Yes

[] Modified Harvard architecture
[] Flexible addressing modes

[] C Compiler optimized instruction set
[] 33 interrupt sources …

28.Jan. 2008

Piri Daniel



```
…

     fractional *p_real = &sigCmpx[0].real; //"fractional" pointer to first of input.real
     fractcomplex *p_cmpx = &sigCmpx[0];    //            - || -            of input.complex


     while(1){

// 1.) SAMPLING
          for (i=0; i<FFT_BLOCK_LENGTH; i++){
               sigCmpx[i].real = ADC10read(0);
               //read analogue for each sample: 0V ... 0x0000, 5V ... 0x0FFF (4095)
               while (TMR1 < 0x0271);  //wait 31.5 us. Refer to dsPICcalc.xls!
          }

// 2.) Sending sample buffer on UART
     …

// 3.) SCALING
     // The FFT function requires input data to be in the fractional fixed-point range [-0.5,
+0.5]
          for (i=0; i<FFT_BLOCK_LENGTH; i++){
               *p_real = *p_real >>1 ; // So, we shift all data samples by 1 bit to the right.
               *p_real++;              // Should you desire to optimize this process,
               //perform data scaling when first obtaining the time samples or within the
               BitReverseComplex function.
          }

// 4.) CLEARING complex parts
          for (i=FFT_BLOCK_LENGTH; i>0; i--){
                    // Convert the Real input sample array to a Complex input sample array
               (*p_cmpx).real = (*p_real--);
                    // We will simpy write zero to the imaginary  part of each sample
               (*p_cmpx--).imag = 0x0000;
          }
```

```
// 5.) FFT (algorythmus in ASM)
          FFTComplexIP(LOG2_BLOCK_LENGTH, &sigCmpx[0], (fractcomplex *)
__builtin_psvoffset(&twiddleFactors[0]), (int) __builtin_psvpage(&twiddleFactors[0]));

// 6.) BIT reversing:
          //Store output samples in bit-reversed order of their addresses
          BitReverseComplex (LOG2_BLOCK_LENGTH, &sigCmpx[0]);

// 7.) CALCULATING REAL part vector (Z = sqrt( Re^2 + Im^2 ))
          // Compute the square magnitude of the complex FFT output array so we have a Real
          output vetor
          SquareMagnitudeCplx(FFT_BLOCK_LENGTH, &sigCmpx[0], &sigCmpx[0].real);

// 8a.) SEARCH the largest spectral component - AMPLITUDE
          // Find the frequency Bin ( =index into the SigCmpx[] array) that has the largest
energy
          VectorMax(FFT_BLOCK_LENGTH/2, &sigCmpx[0].real, &peakFrequencyBin);

// 8b.) FREQUENCY of the largest spectral component
          // Compute the frequency (in Hz) of the largest spectral component
          peakFrequency = peakFrequencyBin*(SAMPLING_RATE/FFT_BLOCK_LENGTH);

// 9.) REPORT UART
          UARTsendString("Fourier Transform and calculations finished!\n\0");
… truncated here

}//end while
```

Berechnungen in

Excel

MathCAD