ADwin

FFT (Fast Fourier Transform) Library

für ADwin Systeme mit ADSP Prozessor (T9, T10)

Version 4.01

Januar 2004

Wichtiger Hinweis :

- Damit diese Library ordungsgemäss funktioniert, sollte sichergestellt werden, dass (mindestens) mit der ADBasic Version 3.20.01 gearbeitet wird.

Inhaltverzeichnis

FFT

Syntax: *return_value = FFT(real, img, result_real, result_img, help_array1, help_array2, number)*

Parameter:	Typ:	Größe :	Beschreibung :
real:	float array	number	Realteil der Originaldaten
img:	float array	number	Imaginärteil der Originaldaten
result_real:	float array	number*4	Realteil der Fourier transformierten Daten
result_img:	float array	number*4	Imaginärteil der Fourier transformierten Daten
help_array1:	float array	number*4	(Speicher für interne Berechnungen)
help_array2:	float array	number*4	(Speicher für interne Berechnungen)
number:	long		Anzahl der Punkte für die FFT (muß ein Vielfaches der Zahl 2 sein)
return_value:	void		(kein Rückgabewert)

Führt eine Fast Fourier Transformation für ein komplexes Datenarray durch.

- Das Ergebnis der FFT ist nicht normiert (skaliert) auf die Größenordnung der einzelnen Signalkomponenten der Originaldaten. Falls eine Normierung erwünscht ist, muß zusätzlich die Funktion *FFT_scale()* aufgerufen werden.
- Obwohl die transformierten Daten notwendigerweise in Arrays der Größe number*4 abgelegt werden, befinden sich die gültigen Daten in den Array- Elementen von 1 bis number/2. (Die restlichen Elemente werden für interne Berechnungen benötigt).
- Normierung der Frequenz- Achse :
  Falls die Originaldaten aus einem gesampelten Zeitsignal mit der gesamten Abtastzeit von „T_total“ bestehen, ist es wie folgt möglich, die Frequenzachse zu normieren :
  Das erste Element der transformierten Daten- Arrays entspricht der Frequenz 0 Hz.
  Das letzte gültige Element (an position number/2) entspricht der Frequenz
  f = (*number* / 2 – 1) / T_total
  Das i-te Element entspricht der Frequenz = (i – 1) / T_total
  (Bemerkung : Der Faktor (i – 1) kommt daher, daß die Arrays mit dem Element 1 starten und nicht mit 0).
  Im unten aufgeführten Beispiel entspricht dem Element 1024 eine Frequenz von (1024 – 1) / 0.1 s = 10230 Hz.
- Um richtige Ergebnisse zu bekommen, sollten die Frequenzkomponenten der Originaldaten in folgendem Bereich liegen : fmin = Abtastfrequenz / *number*
  fmax = Abtastfrequenz / 2 (wegen Abtasttheorem)

Anwendungsbeispiel (für ADwin- GOLD oder L16) :

Beispieldatei :
C:\ADwin\ADbasic\lib\FFT_doc+demo**FFT_demo.bas**

Beschreibung :
Legt man an den analogen Eingang 1 ein Sinussignal von 1000 Hz so wird man bei DATA_3[101] und DATA_4[101] Maxima vorfinden.

Hinweise:

- Falls *number* sich bei mehreren Aufrufen von *FFT()* nicht ändert, kann man die notwendige CPU-Zeit verringern, indem die Funktionen *FFT_init()* und *FFT_calc()* anstelle von *FFT()* verwendet werden.

Hinweise zu allen Funktionen der FFT- library :

- Das Deklarieren der Arrays im internen Speicher („DM_LOCAL“) verringert den Bedarf an Rechenzeit (ca. 23 ms im Vergleich zu ca. 35 ms bei einer 1024 Punkte FFT beim T9 Prozessor).
- Es ist erforderlich, die FFT- library Funktionen entweder in einem niedrig priorisierten Prozeß oder in dem FINISH Abschnitt oder in dem LOWINIT Abschnitt aufzurufen (FINISH und LOWINIT werden ebenfalls mit niedriger Priorität ausgeführt). Ansonsten würde die FFT eventuell die Kommunikation zum ADBasic System zu lange unterbrechen und damit zu einem Fehler führen.

FFT_mag

Syntax: *return_value = FFT_mag(real, img, result_mag, number2)*

Parameter:	Typ:	Größe :	Beschreibung :
real:	float array	number2	Realteil der komplexen Daten
img:	float array	number2	Imaginärteil der komplexen Daten
result_mag:	float array	number2	Ergebnis : Betrag der komplexen Daten
number2:	long		Anzahl der zu konvertierenden Datenelemente
return_value:	void		(kein Rückgabewert)

Konvertiert ein komplexes Datenarray in ein Array, das die Beträge der einzelnen komplexen Elemente enthält.

- Der Betrag jedes Elements wird errechnet mit der Formel : result_mag[i] = SQRT(real[i]^2 + img[i]^2)
- Da das Resultat von *FFT()* ein komplexes Spektrum ist, bietet sich diese Funktion *FFT_mag()* an, um alternativ das Resultat als Betragsfunktion (zusammen mit der Phasenfunktion *FFT_phase()*) darzustellen.
- Falls diese Funktion auf das Ergebnis von *FFT()* angewendet werden soll, so sollte *number2* = number_FFT_points / 2 sein.

Anwendungsbeispiel (für ADwin- GOLD oder L16) :

Beispieldatei :
C:\ADwin\ADbasic\lib\FFT_doc+demo**FFT_mag_demo.bas**

Beschreibung :
Legt man an den analogen Eingang 1 ein Sinussignal von 1500 Hz so wird man bei DATA_5[151] ein Maximum vorfinden.

Hinweise zu allen Funktionen der FFT- library :

- Das Deklarieren der Arrays im internen Speicher („DM_LOCAL“) verringert den Bedarf an Rechenzeit (ca. 23 ms im Vergleich zu ca. 35 ms bei einer 1024 Punkte FFT beim T9 Prozessor).
- Es ist erforderlich, die FFT- library Funktionen entweder in einem niedrig priorisierten Prozeß oder in dem FINISH Abschnitt oder in dem LOWINIT Abschnitt aufzurufen (FINISH und LOWINIT werden ebenfalls mit niedriger Priorität ausgeführt). Ansonsten würde die FFT eventuell die Kommunikation zum ADBasic System zu lange unterbrechen und damit zu einem Fehler führen.

FFT_scale

Syntax: `return_value = FFT_scale(unscaled, result_scaled, number2)`

Parameter:	Typ:	Größe :	Beschreibung :
unscaled:	float array	number2	(unnormierte) Daten
result_scaled:	float array	number2	Ergebnis: normierte Daten
number2:	long		Anzahl der zu normierenden Datenelemente (muß = number_FFT_points / 2 sein)
return_value:	void		(kein Rückgabewert)

Normiert das Ergebnis einer FFT, da die Funktion *FFT()* nicht auf die Größenordnung der einzelnen Signalkomponenten der Originaldaten normiert (skaliert).

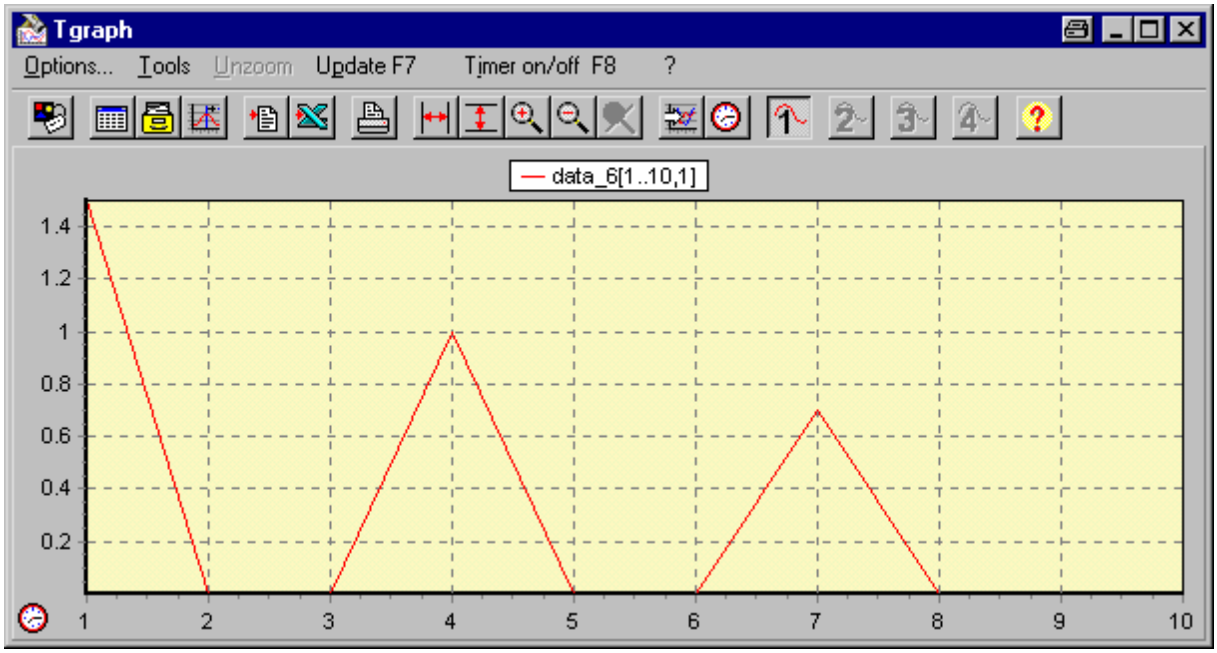
- Diese Funktion arbeitet nach der Formel :
  Für i <> 1 : result_scaled[i] = unscaled[i] / number2
  Für i = 1 : result_scaled[1] = unscaled[1] / (number2 * 2)
- *number2* **muß** = number_FFT_points / 2 sein.
- Diese Funktion normiert NICHT die Frequenzachse des Spektrums (siehe hierzu die Erläuterungen zur Funktion *FFT()*).

Anwendungsbeispiel (für alle ADwin- Systeme mit T9 oder T10 Prozessor) :

Beispieldatei :
C:\ADwin\ADbasic\lib\FFT_doc+demo**FFT_scale_demo.bas**

Beschreibung :
Das Programm generiert intern eine Frequenz von 60 Hz mit der Amplitude von 0.7 (einheitenlos). Dieser Frequenz wird eine Frequenz von 30 Hz mit der Amplitude 1.0 und ein konstanter Offset von 1.5 überlagert. Nach dem Aufruf der Funktionen *FFT(), FFT_mag()* und *FFT_scale()* erhält man ein Spektrum, bei dem die Amplituden der dargestellten Frequenzen exakt den Größen der Signalkomponenten der Originaldaten entsprechen :

DATA_6[7] = 1	(60 Hz)
DATA_6[4] = 0.7	(30 Hz)
DATA_6[1] = 1.5	(konstanter Offset)
DATA_6[alle anderen Elemente] = 0	



Hinweise zu allen Funktionen der FFT- library :

- Das Deklarieren der Arrays im internen Speicher („DM_LOCAL“) verringert den Bedarf an Rechenzeit (ca. 23 ms im Vergleich zu ca. 35 ms bei einer 1024 Punkte FFT beim T9 Prozessor).
- Es ist erforderlich, die FFT- library Funktionen entweder in einem niedrig priorisierten Prozeß oder in dem FINISH Abschnitt oder in dem LOWINIT Abschnitt aufzurufen (FINISH und LOWINIT werden ebenfalls mit niedriger Priorität ausgeführt). Ansonsten würde die FFT eventuell die Kommunikation zum ADBasic System zu lange unterbrechen und damit zu einem Fehler führen.

FFT_phase

Syntax: *return_value = FFT_phase(real, img, result_phase, number2)*

Parameter:	Typ:	Größe :	Beschreibung :
real:	float array	number2	Realteil der komplexen Daten
img:	float array	number2	Imaginärteil der komplexen Daten
result_phase:	float array	number2	Ergebnis : Phase der komplexen Daten
number2:	long		Anzahl der zu konvertierenden Datenelemente
return_value:	void		(kein Rückgabewert)

Konvertiert ein komplexes Datenarray in ein Array, das die Phase der einzelnen komplexen Elemente enthält.

- Die Phase jedes Elements wird errechnet mit der Formel :
  Falls real[i] >= 0 : result_phase[i] = ARCTAN(img[i] / real[i])
  Falls real[i] < 0 : result_phase[i] = ARCTAN(img[i] / real[i]) + *PI*
  (Das detaillierte Listing dieser Formel befindet sich in der Datei : math.inc)
- Da das Resultat von *FFT()* ein komplexes Spektrum ist, bietet sich diese Funktion *FFT_phase()* an, um alternativ das Resultat als Phasenfunktion (zusammen mit der Betragsfunktion *FFT_mag()*) darzustellen.
- Falls diese Funktion auf das Ergebnis von *FFT()* angewendet werden soll, so sollte *number2* = number_FFT_points / 2 sein.

Anwendungsbeispiel (für alle ADwin- Systeme mit T9 oder T10 Prozessor) :

Beispieldatei :
C:\ADwin\ADbasic\lib\FFT_doc+demo**FFT_phase_demo.bas**

Beschreibung :
Dieses Programm generiert intern zwei Frequenzen mit 30 Hz. Die zweite Frequenz hat eine Phasenverschiebung von *PI* / 2 . Beide Signale werden synchron abgetastet. Nach dem für jedes Signal separat die Funktionen *FFT(), FFT_mag(), FFT_scale()* und *FFT_phase()* aufgerufen wurden, erhält man ein Spektrum mit den folgenden Eigenschaften :

DATA_6[4] = 1	(30 Hz)
DATA_6[alle anderen Elemente] = 0	
DATA_7[4] = -0.018410	(Phase = ca. 0)
DATA_26[4] = 1	(30 Hz)
DATA_26[alle anderen Elemente] = 0	
DATA_27[4] = 1.552389	(Phase = ca. *PI* / 2)

Bemerkung : Die Phase aller anderen Elemente ist undefiniert (d. h. ist nicht zwangsweise 0).

Hinweise zu allen Funktionen der FFT- library :

- Das Deklarieren der Arrays im internen Speicher („DM_LOCAL“) verringert den Bedarf an Rechenzeit (ca. 23 ms im Vergleich zu ca. 35 ms bei einer 1024 Punkte FFT beim T9 Prozessor).
- Es ist erforderlich, die FFT- library Funktionen entweder in einem niedrig priorisierten Prozeß oder in dem FINISH Abschnitt oder in dem LOWINIT Abschnitt aufzurufen (FINISH und LOWINIT werden ebenfalls mit niedriger Priorität ausgeführt). Ansonsten würde die FFT eventuell die Kommunikation zum ADBasic System zu lange unterbrechen und damit zu einem Fehler führen.

FFT_mag_scale

Syntax: `return_value = FFT_mag_scale(real, img, result_m_s, number2)`

Parameter:	Typ:	Größe :	Beschreibung :
`real`:	float array	number2	Realteil der komplexen Daten
`img`:	float array	number2	Imaginärteil der komplexen Daten
`result_m_s`:	float array	number2	Erg. : normierter Betrag der komplexen Daten
`number2`:	long		Anzahl der zu konvertierenden Datenelemente
`return_value`:	void		(kein Rückgabewert)

Diese Funktion ist eine Kombination aus *FFT_mag()* und *FFT_scale()* und deshalb schneller als der Aufruf dieser beiden Funktionen.

Anwendungsbeispiel (für alle ADwin- Systeme mit T9 oder T10 Prozessor) :

Beispieldatei :
C:\ADwin\ADbasic\lib\FFT_doc+demo**FFT_scale_demo_opt.bas**

FFT_init

Syntax: `return_value = FFT_init( help_array1, help_array2, number)`

Parameter:	Typ:	Größe :	Beschreibung :
help_array1:	float array	number*4	(Speicher für interne Berechnungen)
help_array2:	float array	number*4	(Speicher für interne Berechnungen)
number:	long		Anzahl der Punkte für die FFT (muß ein Vielfaches der Zahl 2 sein)
return_value:	void		(kein Rückgabewert)

Nur notwendig und sinnvoll bei nachfolgendem Aufruf von *FFT_calc()* oder *FFT_calc_DM()*, etc..
Die Funktion *FFT()* setzt sich zusammen aus den beiden Funktionen *FFT_init()* und *FFT_calc()*.
FFT_init() ist sinnvoll, wenn die FFT mehrmals durchgeführt werden soll. Dann braucht *FFT_init()* nämlich nur ein einziges Mal aufgerufen werden.
Allerdings muss der Parameter *number* bei allen Aufrufen von *FFT_init()* und *FFT_calc()* immer den gleichen Wert haben.

Anwendungsbeispiel (für alle ADwin- Systeme mit T9 oder T10 Prozessor) :

Beispieldatei :
C:\ADwin\ADbasic\lib\FFT_doc+demo**FFT_scale_demo_opt.bas**

FFT_calc

Syntax:
```
return_value = FFT_calc(real, img, result_real, result_img,
                        help_array1, help_array2, number)
```

Parameter:	Typ:	Größe :	Beschreibung :
real:	float array	number	Realteil der Originaldaten
img:	float array	number	Imaginärteil der Originaldaten
result_real:	float array	number*4	Realteil der Fourier transformierten Daten
result_img:	float array	number*4	Imaginärteil der Fourier transformierten Daten
help_array1:	float array	number*4	(Speicher für interne Berechnungen)
help_array2:	float array	number*4	(Speicher für interne Berechnungen)
number:	long		Anzahl der Punkte für die FFT (muß ein Vielfaches der Zahl 2 sein)
return_value:	void		(kein Rückgabewert)

Nur sinnvoll bei vorhergehendem Aufruf von *FFT_init*.
Die Funktion *FFT()* setzt sich zusammen aus den beiden Funktionen *FFT_init()* und *FFT_calc()*.
Sinnvoll, wenn die Fast Fourier Transformation mehrmals durchgeführt werden soll. Dann braucht *FFT_init()* nämlich nur ein einziges Mal aufgerufen werden.

Anwendungsbeispiel (für alle ADwin- Systeme mit T9 oder T10 Prozessor) :

Beispieldatei :
C:\ADwin\ADbasic\lib\FFT_doc+demo**FFT_scale_demo_opt.bas**

FFT_calc_DM

Syntax: `return_value = FFT_calc_DM(real, img, result_real, result_img, help_array1, help_array2, number)`

Parameter:	Typ:	Größe :	Beschreibung :
real:	float array	number	Realteil der Originaldaten
img:	float array	number	Imaginärteil der Originaldaten
result_real:	float array	number*4	Realteil der Fourier transformierten Daten
result_img:	float array	number*4	Imaginärteil der Fourier transformierten Daten
help_array1:	float array	number*4	(Speicher für interne Berechnungen)
help_array2:	float array	number*4	(Speicher für interne Berechnungen)
number:	long		Anzahl der Punkte für die FFT (muß ein Vielfaches der Zahl 2 sein)
return_value:	void		(kein Rückgabewert)

Nur sinnvoll bei vorhergehendem Aufruf von *FFT_init*.
Die Funktion *FFT()* setzt sich zusammen aus den beiden Funktionen *FFT_init()* und *FFT_calc()*.
FFT_calc_DM() ist eine Spezialversion von *FFT_calc()*, die für den T10 Prozessor optimiert ist. Sie kann zwar auch für den T9 Prozessor benutzt werden, hat aber keinen Optimierungseffekt beim T9.
FFT_calc_DM() darf nur verwendet werden, wenn die Arrays im internen Speicher („DM_LOCAL“) deklariert sind. Der Bedarf an Rechenzeit ist ca. 11 ms bei einer 1024 Punkte FFT beim T10 Prozessor, im Vergleich dazu benötigt *FFT_calc()* 14 ms. (Beide Zeitmessungen wurden mit Arrays im DM_LOCAL durchgeführt).

Anwendungsbeispiel (für alle ADwin- Systeme mit T9 oder T10 Prozessor) :

Beispieldatei :
C:\ADwin\ADbasic\lib\FFT_doc+demo**FFT_scale_demo_opt.bas**

FFT_calc_DX

Syntax: `return_value = FFT_calc_DX(real, img, result_real, result_img, help_array1, help_array2, number)`

Parameter:	Typ:	Größe :	Beschreibung :
real:	float array	number	Realteil der Originaldaten
img:	float array	number	Imaginärteil der Originaldaten
result_real:	float array	number*4	Realteil der Fourier transformierten Daten
result_img:	float array	number*4	Imaginärteil der Fourier transformierten Daten
help_array1:	float array	number*4	(Speicher für interne Berechnungen)
help_array2:	float array	number*4	(Speicher für interne Berechnungen)
number:	long		Anzahl der Punkte für die FFT (muß ein Vielfaches der Zahl 2 sein)
return_value:	void		(kein Rückgabewert)

Nur sinnvoll bei vorhergehendem Aufruf von *FFT_init*.
Die Funktion *FFT()* setzt sich zusammen aus den beiden Funktionen *FFT_init()* und *FFT_calc()*.
FFT_calc_DX() ist eine Spezialversion von *FFT_calc()*, die für den T10 Prozessor optimiert ist. Sie kann zwar auch für den T9 Prozessor benutzt werden, hat aber keinen Optimierungseffekt beim T9.
FFT_calc_DX() darf nur verwendet werden, wenn die Arrays im externen Speicher („DRAM_EXTERN“) (= Default) deklariert sind. Der Bedarf an Rechenzeit ist ca. 49 ms bei einer 2048 Punkte FFT beim T10 Prozessor, im Vergleich dazu benötigt *FFT_calc()* 53 ms. (Beide Zeitmessungen wurden mit Arrays im DRAM_EXTERN durchgeführt).

Anwendungsbeispiel (für alle ADwin- Systeme mit T9 oder T10 Prozessor) :

Beispieldatei :
C:\ADwin\ADbasic\lib\FFT_doc+demo**FFT_scale_demo_opt_DX.bas**