BEUTH HOCHSCHULE
FÜR TECHNIK
BERLIN
University of Applied Sciences

Analog-Digital-Umsetzer und digitale Signale

Laborbericht

angefertigt von

Robby Kozok, Nic Frank Siebenborn, Pascal Kahlert

in dem Fachbereich VII – Elektrotechnik - Mechatronik - Optometrie –
für das Modul Digitale Signalverarbeitung III
der Beuth Hochschule für Technik Berlin im Studiengang
Elektrotechnik - Schwerpunkt Elektronische Systeme

Datum 3. November 2015

Lehrkraft
Prof. Dr.-Ing Marcus Purat Beuth Hochschule für Technik

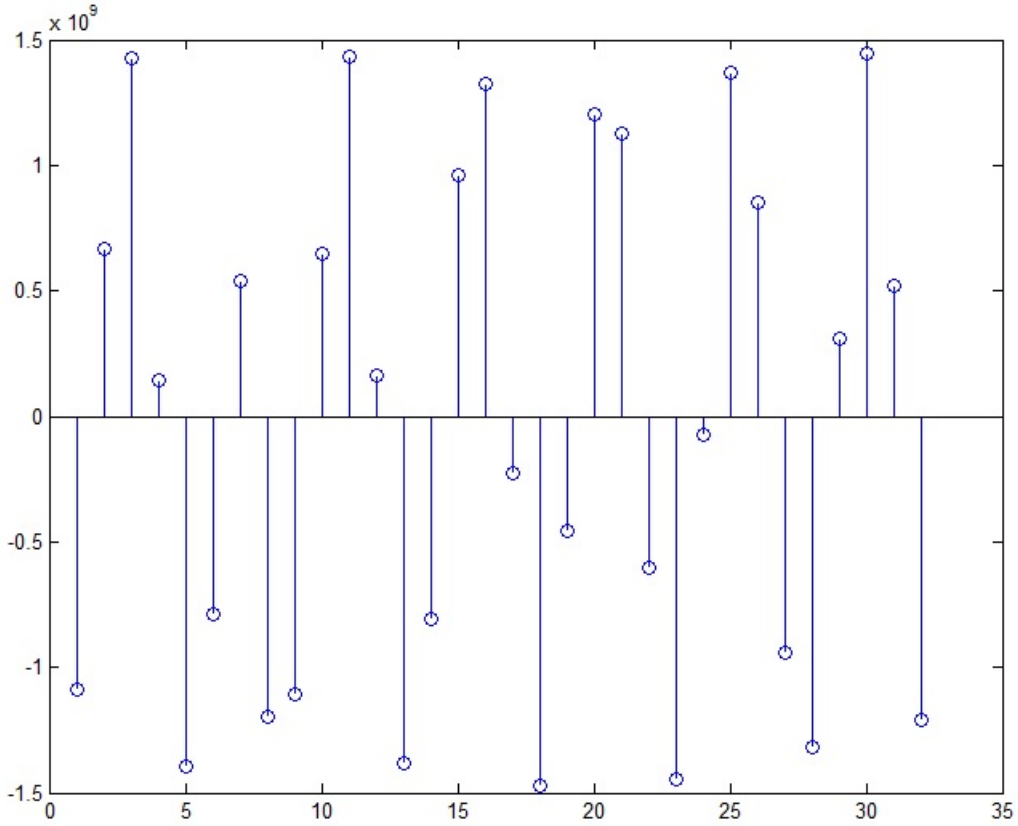
Inhaltsverzeichnis

Kapitel 1

Einlesen von Signalen

```
1  #include "codeclib.h"
2  #include "copydata.h"
3
4
5  /*       @function        copyData
6  *        @brief           Kopiert die Audiodaten des Kanals "Internal ADC R0" aus
7  *                 dem DMA-Lesepuffer in den Speicherbereich iInput schreibt.
8  *        @param           input   Adresse der ersten Speicherstelle von input
9  *        @param           pWrite  Zeiger auf die Adresse von input
10 *        @param           size    Anzahl der übergebenen Werte
11 *        @return          void
12 */
13 void copyData(int *input, int **pWrite, int size) {
14
15         // Nimm den 5ten Wert aus iDMARxBuffer und schreibe diesen in den input.
16         **pWrite = iDMARxBuffer[4];
17
18         // Inkrementiere den Wert der zuletzt beschriebenen Adresse
19         (*pWrite)++;
20
21         // Wenn die obere Grenze size erreicht wurde müssen wir auf die
22         // Startadresse zurückspringen.
23         if(*pWrite == input + size)
24         {
25                 *pWrite = input;
26         }
27
28 }
```

Hier zu sehen Bilder, wow!


x 10^9
1.5
1
0.5
0
-0.5
-1
-1.5
0
5
10
15
20
25
30
35

Kapitel 2

Ausgeben von Signalen

Kapitel 3

Verarbeiten von Signalen

Anhang A

Quelltext-Dateien

```c
#include "codeclib.h"
#include "copydata.h"


/*      @function       copyData
*       @brief          Kopiert die Audiodaten des Kanals "Internal ADC R0" aus dem
*                               in den Speicherbereich iInput schreibt.
*       @param          input   Adresse der ersten Speicherstelle von input
*       @param          pWrite  Zeiger auf die Adresse von input
*       @param          size    Anzahl der übergebenen Werte
*       @return         void
*/
void copyData(int *input, int **pWrite, int size) {

        //Nimm den 5ten Wert aus iDMARxBuffer und schreibe diesen in den input.
        **pWrite = iDMARxBuffer[4];

        //Inkrementiere den Wert der zuletzt beschriebenen Adresse
        (*pWrite)++;

        // Wenn die obere Grenze size erreicht wurde müssen wir auf die Startadresse
        if(*pWrite == input + size)
        {
                *pWrite = input;
        }

}
```

copydata.c