

*Application Note
Interfacing a PCM echo
Canceller to the GR47*

First edition (October 2003)

Sony Ericsson Mobile Communications. publishes this manual without making any warranty as to the content contained herein. Further **Sony Ericsson Mobile Communications.** reserves the right to make modifications, additions and deletions to this manual due to typographical errors, inaccurate information, or improvements to programs and/or equipment at any time and without notice. Such changes will, nevertheless be incorporated into new editions of this manual.

All rights reserved.

© **Sony Ericsson Mobile Communications.**, 2003

Contents

| | | |
|----------|------------------------------------|----------|
| 1 | INTRODUCTION | 4 |
| 2 | DESIGN OVERVIEW | 5 |
| 2.1 | GENERAL..... | 5 |
| 2.2 | SPI CONTROL..... | 5 |
| 3 | OPERATION | 6 |
| 3.1 | GENERAL..... | 6 |
| 3.2 | APPLICATION REGISTER SETTINGS..... | 7 |
| 3.3 | EMBEDDED APPLICATION CODE | 7 |

1 Introduction

The GR47 has been designed primarily for use with a headset or handset for audio communication. If the GR47 is to be used in a speakerphone environment where a loudspeaker and external microphone replace the headset there may be much acoustic coupling of the inbound speech into the microphone path. The result is a high degree of echo observed by the far-end party. The party speaking at the GR47-end does not observe any such echo. This application note provides a solution for echo-cancellation when using the GR47 in this manner

2 Design Overview

2.1 General

This echo-cancelling application uses the digital audio interface to the GR47. A block diagram of the GR47 audio system is shown in Figure 1. The echo-cancellation is performed by a digital audio IC from Zarlink Semiconductor (part number MT93L16AQ) which is inserted into the digital audio link between the GSM DSP and the audio Codec as shown in Figure 2.

The digital audio PCM connections for the GR47 and the MT93L16AQ are similar, but not identical. The GR47 uses a 16 bit linear PCM word with a short SYNC pulse triggering the start of a data word. The MT93L16AQ uses a 16 bit linear PCM word (in SSI mode), but in order to differentiate between 8 bit and 16 bit, the SYNC pulse must last for the entire length of the 16 bit word. In order to satisfy the requirements of the MT93L16AQ SYNC pulse, some extra logic is required between the devices lengthen the SYNC pulse from the GR47. The circuit design to achieve this is shown in Figure 3.

Note! For the SYNC extending circuit to operate correctly it is important that the clock and SYNC edges be properly sequenced. Rising edge of CLK must trigger the counter before the rising edge of SYNC. The use of NOR gates as shown inherently provides the necessary sequencing due to gate delays on the SYNC signal. Simple NAND gate substitution will add gate delay into the CLK signal and may cause a race condition hazard. It may be acceptable to introduce a small RC delay onto the SYNC signal if encountering problems.

2.2 SPI Control

The MT93L16AQ is controlled by an SPI bus. The bus is compatible with the SPI intrinsic functions of the GR47 M2mpower solution. The GR47 pins used for SPI control are:

| GR47 PIN # | PIN NAME | SIGNAL | DESCRIPTION |
|------------|----------|----------|---|
| 21 | IO1 | CLOCK | SPI Clock – Data Latched on Rising Edge |
| 22 | IO2 | DATA IN | Data from the external peripheral into the GR47 |
| 23 | IO3 | DATA OUT | Data to the external peripheral from the GR47 |
| 36 | RI | SYNC | Defines start of data. Asserted for entire data send time |

3 Operation

3.1 General

The MT93L16AQ contains two echo cancellers, one for acoustic echo cancelling and the other for line echo cancelling.

As a result of the high levels of signal gain required in the microphone path for hands-free applications it has not been possible to use the IC in its normal manner, i.e. using the acoustic echo path (Port 2). Instead it has been necessary to utilise the signal gain of the receive path (Register 20h) and reduce the gain in the GR47 Codec. This keeps the output and input signal within acceptable levels for cancellation.

The result is that the microphone-speaker echo is introduced into the Line Echo circuit (Port 1) instead of the acoustic echo circuit (Port 2). It has been found that the MT93L16AQ operates acceptably in this 'reverse' configuration due to the symmetrical nature of the internal adaptive filter blocks. However the filter cancellation time will be limited to only 16ms in this configuration instead of the 112mS which would have been achievable through Port 2.

Therefore, there is a limitation of the size of room in which the presented design will operate successfully. The echo path-length for 16ms is approximately 5m. This will be sufficient for most car-kit applications, and speaker-phone applications in a small room or where the predominant echo is from close coupling of microphone to speaker rather than through room reverberation.

Note! It may be possible for an extra digital gain stage to be inserted into the microphone path after the echo cancellation has been performed. This could allow the designer to return the MT93L16AQ to its preferred orientation. This is not presented here in order to minimise component count in the proposed solution.

The MT93L16AQ echo-canceller uses internal registers to set its functionality. Once configured, it constantly monitors the signals on the send and receive paths measuring the correlation between outgoing speech and any returned echo. The adaptive filter constantly adjusts to the correlation between signal and echo allowing the echo canceller to track changes in the echo path.

3.2 Application Register Settings

Following the default reset conditions the following register settings were altered for the application presented.

| Register | Setting | Description |
|----------|---------|--|
| 00h | 04h | Main Control Register – Switch off AGC |
| 20h | 6Fh | Receive Gain Control – Set to +21dB |
| 21h | 00h | Acoustic Echo Canceller Control – Enable |
| 01h | 00h | Line Echo Canceller Control – Enable |
| 24h | 80h | Rout Limiter 1 – Maximum |
| 25h | FFh | Rout Limiter 2 – Maximum |
| 26h | F8h | Sout Limiter - Maximum |

3.3 Embedded Application Code

The following code sample can be used to set-up the MT93L16AQ with the register settings provided in table xx.

```

/* CONSTANTS */
char MCrege = 0x00;
char LECreg = 0x01;
char RGCreg = 0x20;
char AECreg = 0x21;
char RL1reg = 0x24;
char RL2reg = 0x25;
char SLreg = 0x26;
char READ = 0x80;
char WRITE = 0x00;

/* Global Variables */
char SPIbuf[10];

main()
{
    prs(0);
    printf("\n ##### Script Started #####");
    spic (1,1,2); /* enable SPI, sync pulse low for 2 bytes. */
    MT93L16_Init();
    spic (0,0,0); /* disables SPI */
    printf("\n ##### Script Completed #####");
}

MT93L16_Init()
{
    /* Main Control Register 0x00 */
    SPIbuf[0] = (MCrege << 1) | WRITE;

```

```

SPIbuf[1] = 0x04;
printf("\n SPI data out : %x %x", SPIbuf[0], SPIbuf[1]);
spis(SPIbuf, 2);
SPIbuf[0] = (MCreg << 1) | READ;
SPIbuf[1] = 0xff;
printf("\n SPI data out: %x %x", SPIbuf[0], SPIbuf[1]);
spis(SPIbuf, 2);
printf("\n SPI data in: %x %x", SPIbuf[0], SPIbuf[1]);
dlyms(1);

/* Gain Control Register 0x20 */
SPIbuf[0] = (RGCrege << 1) | WRITE;
SPIbuf[1] = 0x6f;
printf("\n SPI data out : %x %x", SPIbuf[0], SPIbuf[1]);
spis(SPIbuf, 2);
SPIbuf[0] = (RGCrege << 1) | READ;
SPIbuf[1] = 0xff;
printf("\n SPI data out: %x %x", SPIbuf[0], SPIbuf[1]);
spis(SPIbuf, 2);
printf("\n SPI data in: %x %x", SPIbuf[0], SPIbuf[1]);
dlyms(1);

/* Acoustic Canceller Control Register 0x21 */
SPIbuf[0] = (AECreg << 1) | WRITE;
SPIbuf[1] = 0x00;
printf("\n SPI data out : %x %x", SPIbuf[0], SPIbuf[1]);
spis(SPIbuf, 2);
SPIbuf[0] = (AECreg << 1) | READ;
SPIbuf[1] = 0xff;
printf("\n SPI data out: %x %x", SPIbuf[0], SPIbuf[1]);
spis(SPIbuf, 2);
printf("\n SPI data in: %x %x", SPIbuf[0], SPIbuf[1]);
dlyms(1);

/* Line Canceller Control Register 0x01 */
SPIbuf[0] = (LECrege << 1) | WRITE;
SPIbuf[1] = 0x00;
printf("\n SPI data out : %x %x", SPIbuf[0], SPIbuf[1]);
spis(SPIbuf, 2);
SPIbuf[0] = (LECrege << 1) | READ;
SPIbuf[1] = 0xff;
printf("\n SPI data out: %x %x", SPIbuf[0], SPIbuf[1]);
spis(SPIbuf, 2);
printf("\n SPI data in: %x %x", SPIbuf[0], SPIbuf[1]);
dlyms(1);

/* Sout Limiter Register 0x26 */
SPIbuf[0] = (SLreg << 1) | WRITE;
SPIbuf[1] = 0xf8;
printf("\n SPI data out : %x %x", SPIbuf[0], SPIbuf[1]);
spis(SPIbuf, 2);
SPIbuf[0] = (SLreg << 1) | READ;

```

```

SPIbuf[1] = 0xff;
printf("\n SPI data out: %x %x", SPIbuf[0], SPIbuf[1]);
spis(SPIbuf, 2);
printf("\n SPI data in: %x %x", SPIbuf[0], SPIbuf[1]);
dlyms(1);

/* Rout Limiter '2' Register 0x25 */
SPIbuf[0] = (RL2reg << 1) | WRITE;
SPIbuf[1] = 0xff;
printf("\n SPI data out : %x %x", SPIbuf[0], SPIbuf[1]);
spis(SPIbuf, 2);
SPIbuf[0] = (RL2reg << 1) | READ;
SPIbuf[1] = 0xff;
printf("\n SPI data out: %x %x", SPIbuf[0], SPIbuf[1]);
spis(SPIbuf, 2);
printf("\n SPI data in: %x %x", SPIbuf[0], SPIbuf[1]);
dlyms(1);

/* Rout Limiter '1' Register 0x24 */
SPIbuf[0] = (RL1reg << 1) | WRITE;
SPIbuf[1] = 0x80;
printf("\n SPI data out : %x %x", SPIbuf[0], SPIbuf[1]);
spis(SPIbuf, 2);
SPIbuf[0] = (RL1reg << 1) | READ;
SPIbuf[1] = 0xff;
printf("\n SPI data out: %x %x", SPIbuf[0], SPIbuf[1]);
spis(SPIbuf, 2);
printf("\n SPI data in: %x %x", SPIbuf[0], SPIbuf[1]);
dlyms(1);
}

```

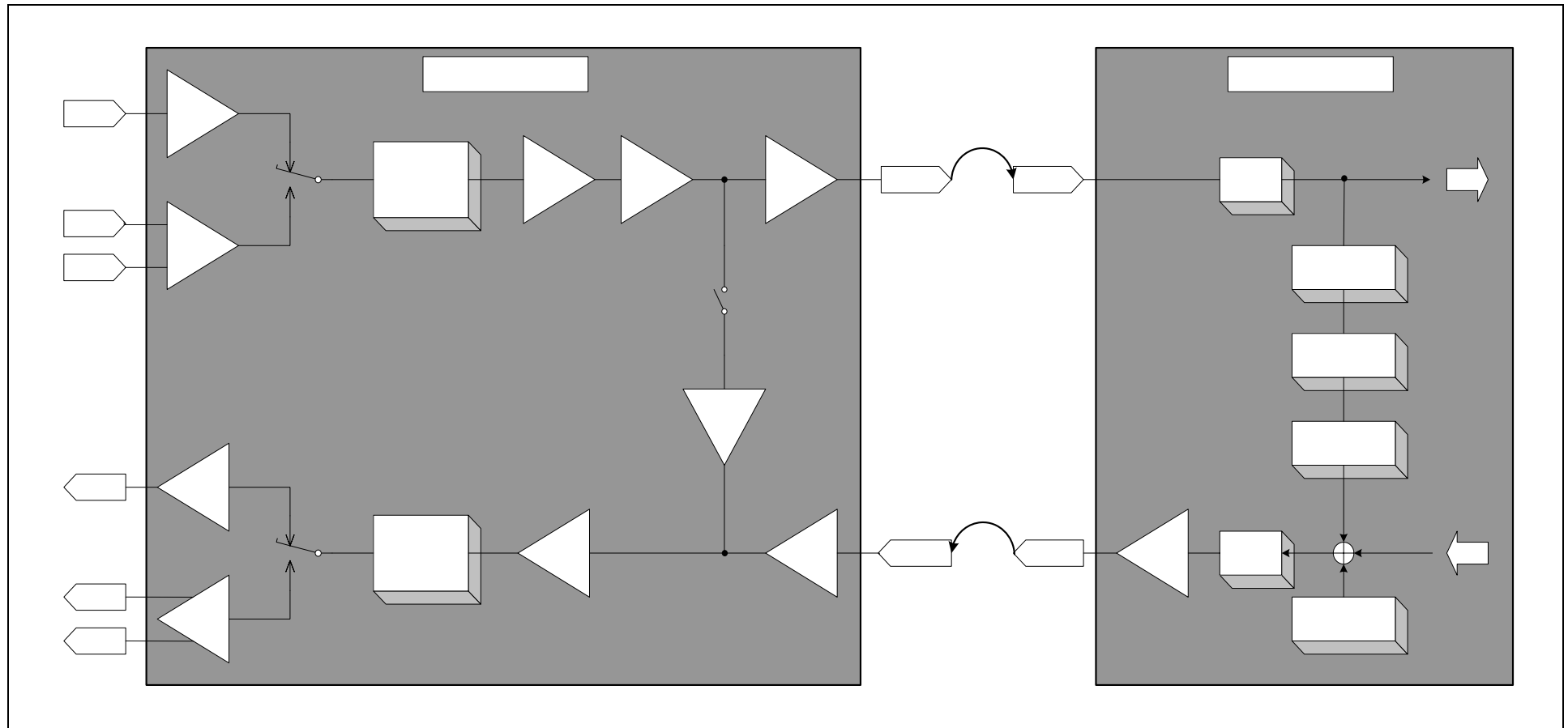


Figure 1 - GR47 Audio Block Diagram
 ATMS
 AuxGain
 N=4

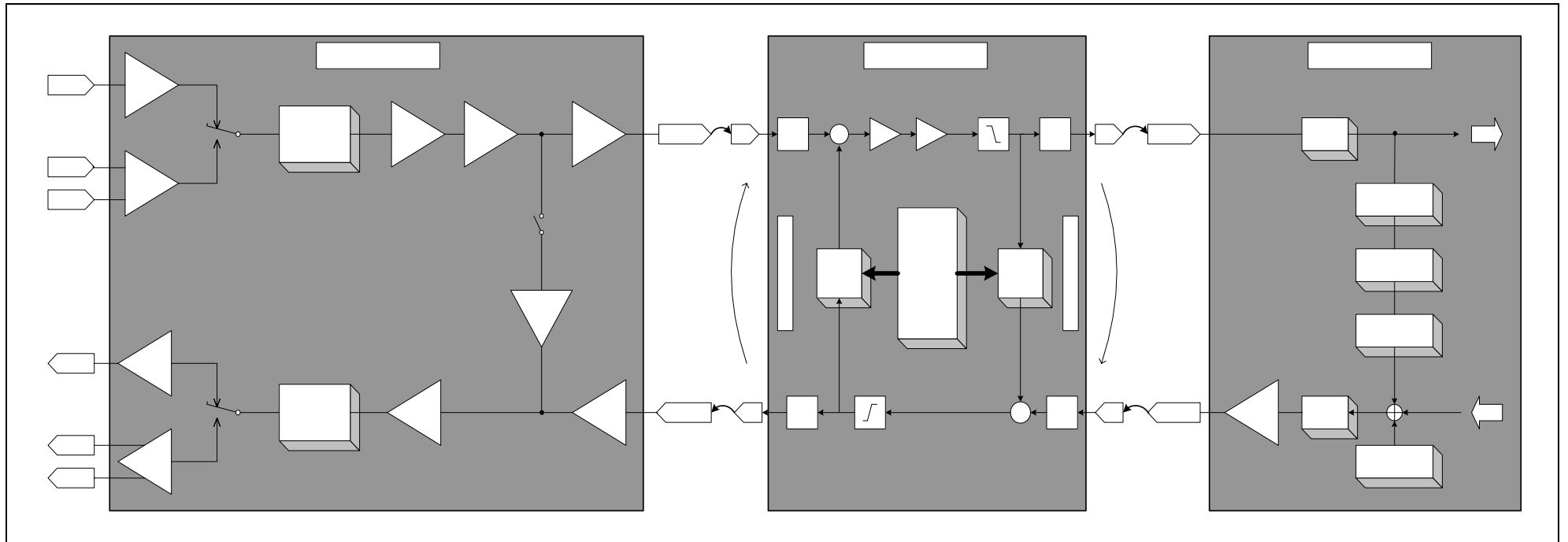


Figure 2. Echo-Canceller Pathway

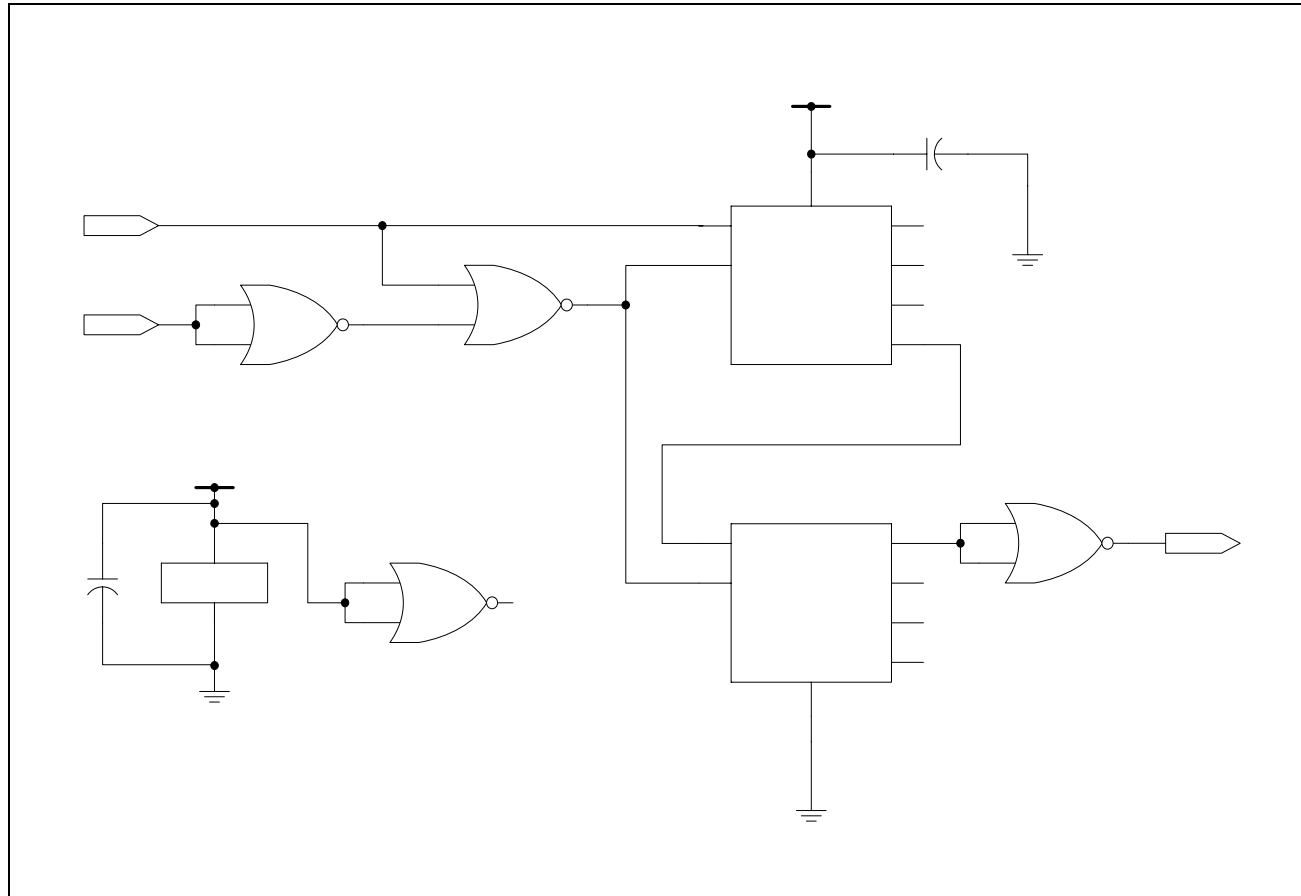


Figure 3 - PCMSYN to SSISYNC Conversion

PCMCLK