Application Report

SBAA105 – October 2003

Clock Divider Circuit for the ADS1202 in Mode 3 Operation

Miroslav Oljaca, Herbert Braisz *Data Acquisition Group*

ABSTRACT

The ADS1202 is a precision, 80dB dynamic range, delta-sigma (ΔΣ) modulator operating from a single +5V supply. The differential inputs are ideal for direct connections to transducers or low-level signals. With the appropriate digital filter and modulator rate, the device can be used to achieve 15-bit Analog-to-Digital (A/D) conversion with no missing codes. This application note describes how to operate the ADS1202 in Mode 3 with an external clock.

Contents

Figures

1 Introduction

Figure 1 shows the block diagram of the ADS1202. The second-order delta-sigma modulator can receive a clock signal from either the internal 20MHz oscillator or an external clock source. The internal oscillator is enabled when the ADS1202 operates in Mode 0, 1 or 2. The output signals in these modes are MCLK and MDAT. The output data, MDAT, is synchronized and read by an external device, referred to as MCLK. Figure 2 presents the flexible digital interface block diagram.

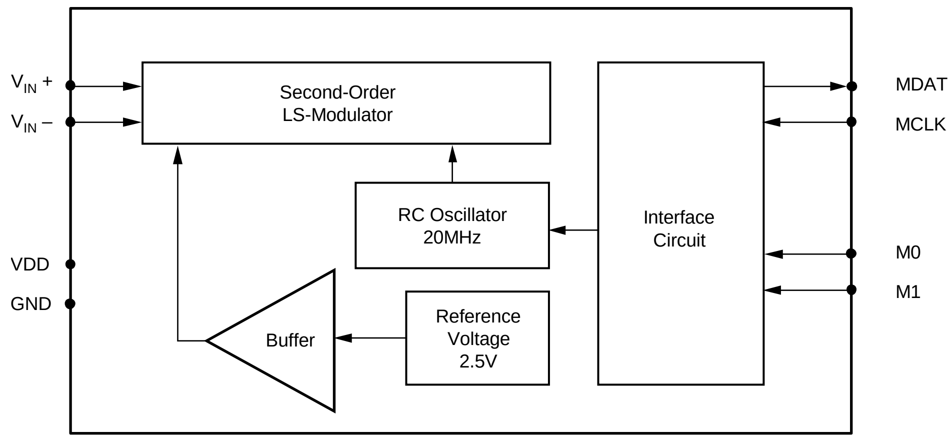


Figure 1. ADS1202 Block Diagram

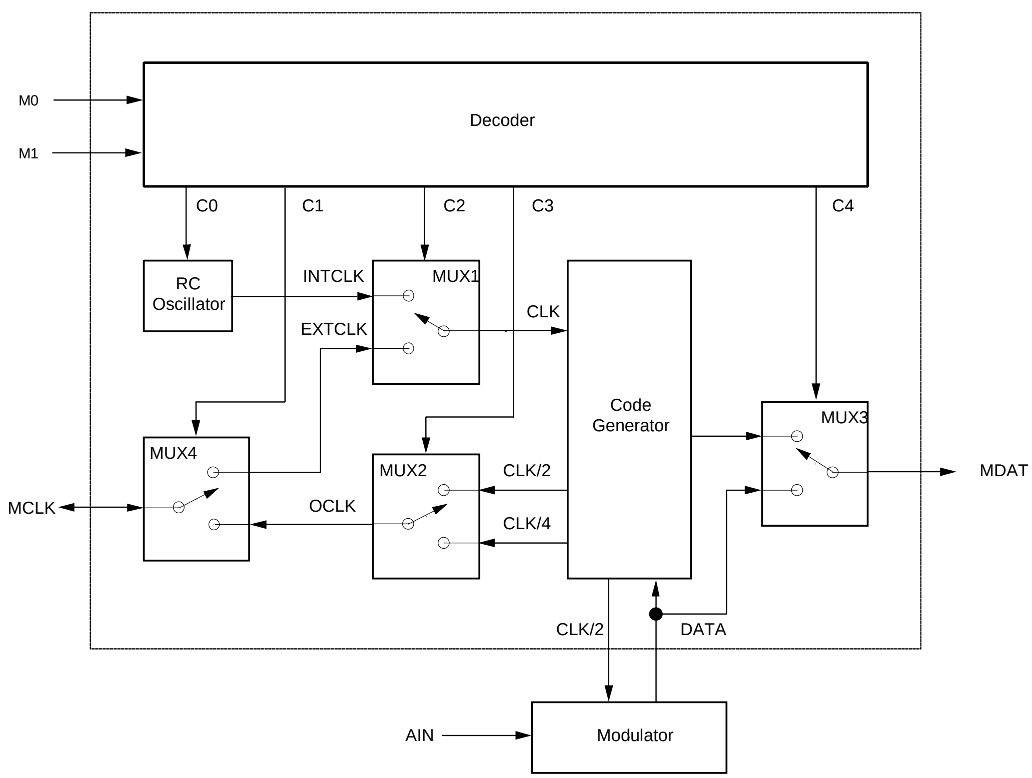


Figure 2. Flexible Interface Block Diagram

Figure 2 shows two mode selector input (or control) signals, M0 and M1. The input signals enter the flexible interface circuit via the decoder, which decodes the input codes and selects the desired mode of operation. With two control lines, M0 and M1, it is possible to select four different operating modes: Mode 0, Mode 1, Mode 2, and Mode 3, which are shown in Table 1.

Table 1. Definition and Description of ADS1202 Operating Modes

MODE	DEFINITION	M1	M0
0	Internal Clock, output 10MHz Synchronous Data Output 10MB (5MHz)	LOW	LOW
1	Internal Clock, output 5MHz Synchronous Data Output 10MB (5MHz)	LOW	HIGH
2	Internal Clock, no output Manchester Coded Data Output 20MB (10MHz)	HIGH	LOW
3	External Clock, input from 500kHz to 20MHz Synchronous Data Output one-fourth of input frequency	HIGH	HIGH

When several ADS1202 devices are operating in parallel, or when an external clock is preferred, the ADS1202 can be set up to operate in Mode 3. The internal oscillator will be disabled and the external clock is applied to the MCLK pin. The ADS1202 divides this clock internally, and uses only half of the supplied clock frequency for data conversion. This division is done to insure a 50% duty cycle of the conversion clock for best performance. The external circuit is required to synchronize with this new clock. In this application note, we describe the circuit needed to selectively divide and synchronize the input clock with the sampling clock of the ADS1202, providing the necessary signals for reading converter output data.

2 ADS1202 Timing

The clock divider interface (CDI) circuit will interface to the ADS1202 operating in Mode 3; on the output side, the CDI circuit provides signals that are equivalent to the signals when the ADS1202 is operating in Mode 1. The CDI circuit configuration is presented in Figure 3. The oscillator provides the main clock for the complex programmable logic device (CPLD). The CPLD supplies the signal MCLK to the ADS1202. MDATA is a data signal coming from the modulator. The two signals MCLKm and MDATm are supplied by the CPLD to the control electronics.

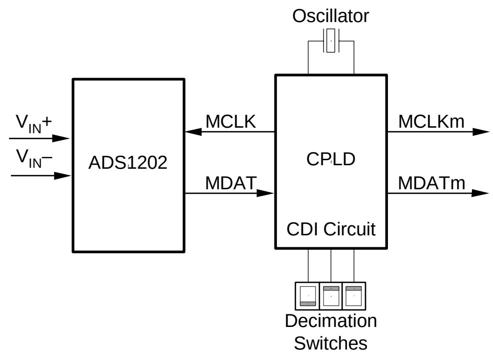


Figure 3. CDI Circuit Interfacing ADS1202

When the ADS1202 operates in Mode 1, the input signal M0 is high and M1 is low (see Table 1). Referring to Figure 2, the first control signal (C0) coming from the mode decoder logic enables the ADS1202 internal RC oscillator that provides the clock signal INTCLK as an input to MUX1. The second control signal (C2) coming from the decoder positions MUX1 so that the output signal becomes the input signal to the code generator, INTCLK. The output signal from the delta-sigma modulator, DATA, is also the MDAT signal coming from the modulator, because the control signal (C4) from the decoder positions MUX3 for that operation. MUX2 is positioned for the mode directed by the control signal (C3) coming from the decoder with an OCLK of CLK/2. The output clock signal MCLK comes through MUX4 (control signal C1) from MUX2 as OCLK or CLK/2. The signal timings for mode 1 operation are presented in Figure 4. In this mode, the control circuit reads data on every edge, rising and falling, of the output clock.

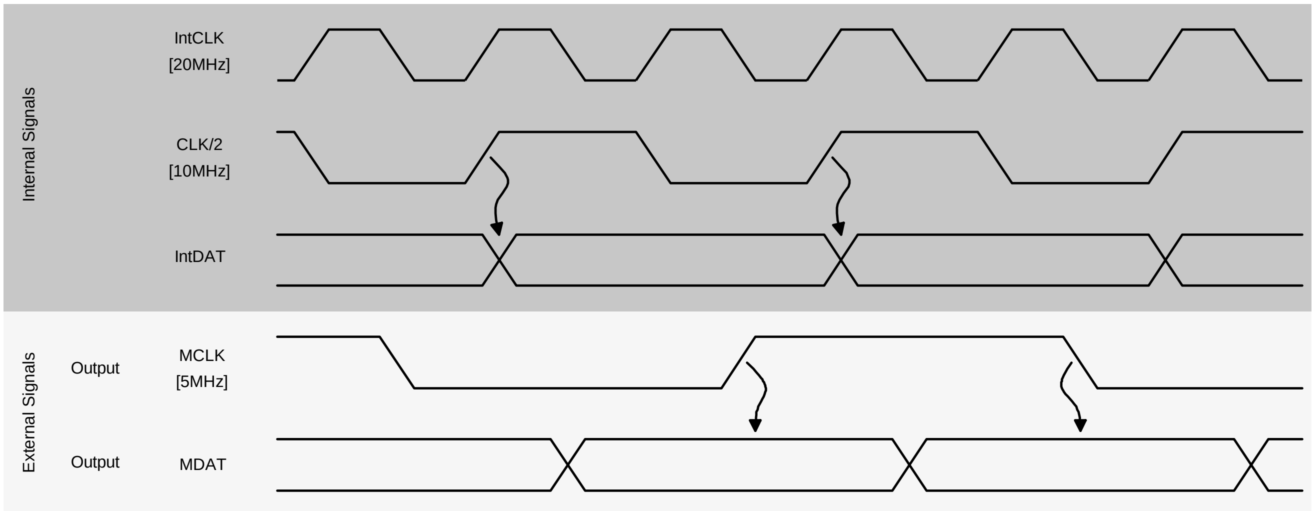


Figure 4. ADS1202 Output Waveform in Mode 1

Mode 3 is similar to Mode 0; the only difference is that an external clock (EXTCLK) is provided. In Mode 3, both input signals M0 and M1 are HIGH (see Table 1). The control signal (C0) coming from the decoder disables the internal RC oscillator. The port MCLK is used as an input for the external clock EXTCLK. The multiplexer MUX4 is directed by control signal (C1) to provide the signal EXTCLK as an input to MUX1. The control signal (C2) coming from the decoder positions MUX1 so that the output signal CLK is equal to EXTCLK. The output signal MDAT is the DATA signal coming directly from the delta-sigma modulator, because the control signal (C4) from the decoder positions MUX3 for that operation. The signal timings for Mode 3 operation are presented in Figure 5. In this mode, the control circuit reads data on every second falling edge of the input clock.

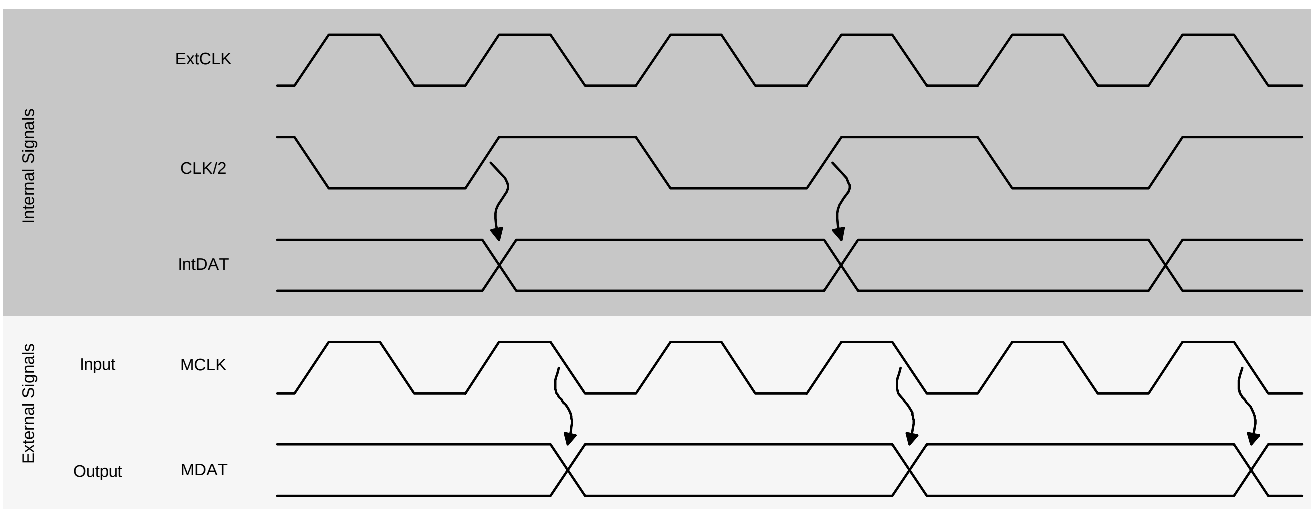


Figure 5. ADS1202 Output Waveform in Mode 3

3 CDI Circuit

As previously mentioned, operation of the ADS1202 in Mode 3 is recommended for some applications. By using an external clock, the noise is minimized, and it is possible to obtain the best performance from the ADS1202. In a configuration where the CDI circuit is used (see Figure 6), an external clock (MCLK) that is used to obtain the desired sampling and output frequency replaces the ADS1202 internal oscillator. The three input lines (M0, M1, and M2) into the CPLD will decode a factor by which the oscillator frequency will be divided before being provided to the ADS1202 MCLK pin. With the MCLK signal and the MDAT provided by the ADS1202 (see the timing diagram in Figure 5), the CDI generates the signals MCLKm and MDATm with the timing shown in Figure 7. The following sections describe the function of the CDI circuit modules.

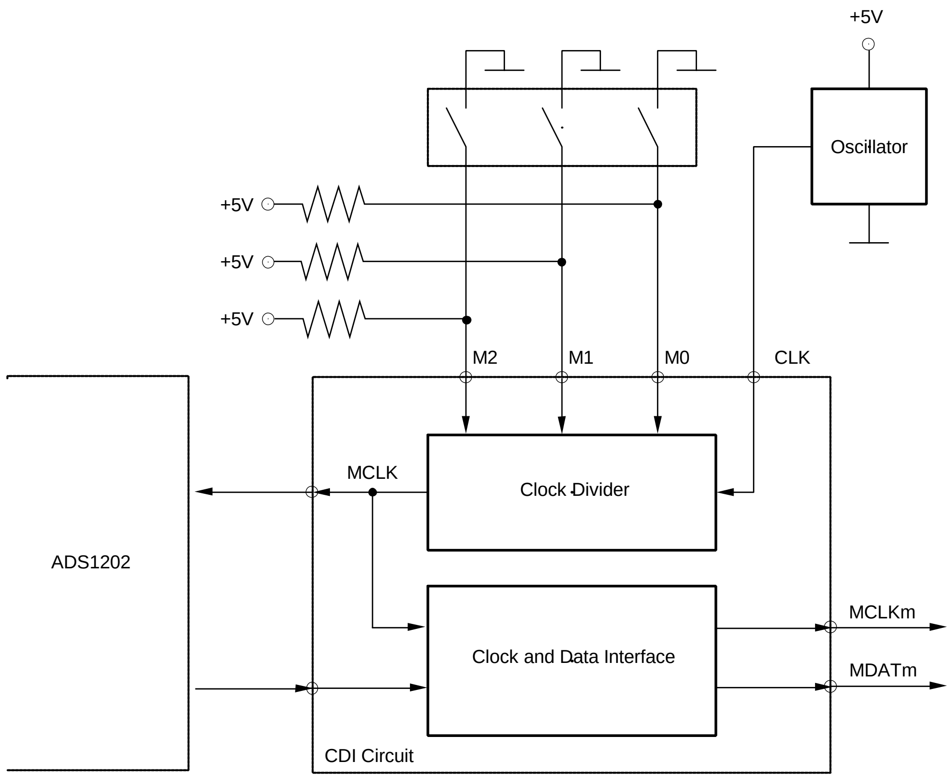


Figure 6. Structure of the CDI Circuit

3.2 Clock Divider

The Clock Divider module generates the clock signal (MCLK) for the ADS1202. Depending on the switch selection (M2, M1, or M0) the input clock CLK will be divided. Table 2 shows the different output frequencies for each switch selection as well as the corresponding oscillator frequency.

Table 2. Different Output Frequency Selection

Switch selection				Oscillator frequency [MHz]				
M2	M1	M0	N	20	18	16	14	12
0	0	0	1	20.0	18.0	16.0	14.0	12.0
0	0	1	2	10.0	9.0	8.0	7.0	6.0
0	1	0	3	6.7	6.0	5.3	4.7	4.0
0	1	1	4	5.0	4.5	4.0	3.5	3.0
1	0	0	5	4.0	3.6	3.2	2.8	2.4
1	0	1	6	3.3	3.0	2.7	2.3	2.0
1	1	0	7	2.9	2.6	2.3	2.0	1.7
1	1	1	8	2.5	2.3	2.0	1.8	1.5

When the input oscillator clock is divided, the signal MCLK is supplied from the CPLD to the ADS1202. The internal delta-sigma modulator will use that signal, and after conversion will provide output data MDAT. The MDAT signal coming from the ADS1202 will have valid data that can be read by the control logic on every second falling edge of the incoming signal, MCLK, as shown in Figure 5.

3.3 Clock and Data Interface

A second function of the CDI circuit is to create the clock output MCLKm synchronously with data (MDAT) coming from the ADS1202. The synchronized data output is called MDATm. The routine will wait for the change of the incoming signal MDAT. When a change in the status of the MDAT signal occurs, the next falling edge of MCLK will be the edge when the output signal MCLKm changes state. If, after two following MCLK cycles, a MDAT change does *not* happen, the MCLKm will continue to change state. The output signal will continue to run synchronously at MCLK frequency. As MDAT, MDATm, MCLK and MCLKm are now synchronized, the output MCLKm will be resynchronized with every new change in the incoming MDAT signal.

3.4 CDI Circuit Implementation

The two required functions described above, *Clock Divider* and *Clock and Data Interface*, are implemented in a CPLD. The structure of the CDI circuit is shown in Figure 6 and the VHDL code that was used is printed in Appendix A.

Figure 7 shows the relationship between the input and output signals of the CDI. It shows the input and output signal to the ADS1202 as well as output signals to the control circuit.

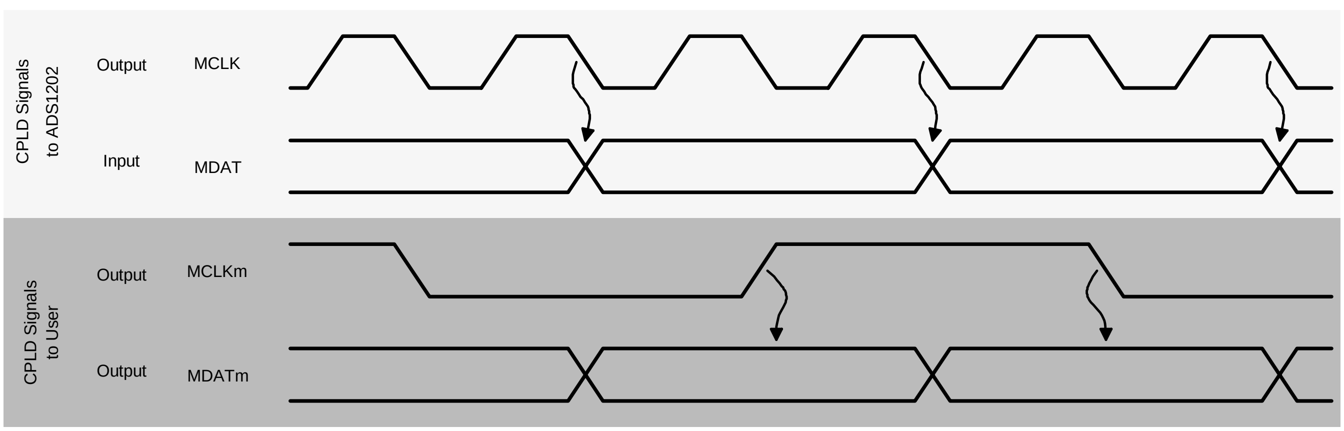


Figure 7. Input and Output Waveforms from CDI Circuit

The schematic for the board that was used to obtain these results is given in Appendix B.

Conclusion

This example provides a step-by-step process in specifying and designing a CDI circuit. It shows that with a simple CPLD, it is possible to build evaluation features that will permit evaluation of an ADS1202 that is operating in Mode 3. This evaluation feature is especially useful when an application requires that multiple ADS1202 devices operate synchronously from the same external clock source. Another feature of this example is that the ADS1202 under investigation can be supplied with different clock frequencies from 500kHz up to 20MHz. In this manner, the performance of the overall system at different frequencies can be compared, and an optimum configuration can be chosen.

APPENDIX A.

Implemented VHDL code for CDI circuit functions:

```
library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.std_logic_unsigned.all;
entity Clock_Divider is
   port (M0, M1, M2 : in std_logic;
         CLK, resn  : in std_logic;
         MCLK       : out std_logic;
         MCLKn      : out std_logic
   );
end Clock_Divider;

architecture RTL of Clock_Divider is
   signal div_reg, counter, counter_n : std_logic_vector (2 downto 0);
   signal counter_exp, MCLK_int, MCLK_temp, CLKn : std_logic;
   signal Mx_sel : std_logic_vector (2 downto 0);
   signal cnt_en, cnt_en_n : std_logic;
   signal div_reg_load, div_reg_load_reset : std_logic;
begin

   Mx_sel <= M2 & M1 & M0;
   CLKn <= not CLK;

   process(CLKn, resn)
   begin
      if resn = '0' then
         div_reg_load        <= '0';
         div_reg_load_reset  <= '0';
      elsif CLKn'event and CLKn = '1' then
         if div_reg_load_reset = '1' then
            div_reg_load <= '0';
         else
            div_reg_load <= '1';
            div_reg_load_reset   <= '1';
         end if;
      end if;
   end process;

   process(CLKn, resn)
   begin
      if resn = '0' then
         div_reg <= "000";
      elsif CLKn'event and CLKn = '1' then
         if div_reg_load = '1' then
            div_reg <= Mx_sel;
         end if;
      end if;
   end process;
```

```
process(CLK, resn)
begin
   if resn = '0' then
      counter <= "000";
   elsif CLK'event and CLK = '1' then
      if counter = "000" then
         counter <= div_reg;
      else
         counter <= counter - 1;
      end if;
   end if;
end process;
process(CLKn, resn)
begin
   if resn = '0' then
      counter_n <= "000";
   elsif CLKn'event and CLKn = '1' then
      counter_n <= counter;
   end if;
end process;

process(counter, div_reg)
begin
   cnt_en <= '0';
   case div_reg is
     when "000" => cnt_en <= '1';               -- divide by 1
     when "001" => if counter = "001" then      -- divide by 2
                      cnt_en <= '1';
                   end if;
     when "010" => if counter = "010" or counter = "001" then   -- divide by 3
                      cnt_en <= '1';
                   end if;
     when "011" => if counter = "011" or counter = "010" then   -- divide by 4
                      cnt_en <= '1';
                   end if;
     when "100" =>
        if counter = "100" or counter = "011" or counter = "010" then -- divide by 5
           cnt_en <= '1';
        end if;
     when "101" =>
        if counter = "101" or counter = "100" or counter = "011" then -- divide by 6
           cnt_en <= '1';
        end if;
     when "110" =>
        if counter = "110" or counter = "101" or counter = "100" or
              counter = "011" then   -- divide by 7
           cnt_en <= '1';
        end if;
     when "111" =>
        if counter = "111" or counter = "110" or counter = "101" or
              counter = "100" then   -- divide by 8
           cnt_en <= '1';
        end if;
     when others => cnt_en <= '1';
   end case;
end process;
```

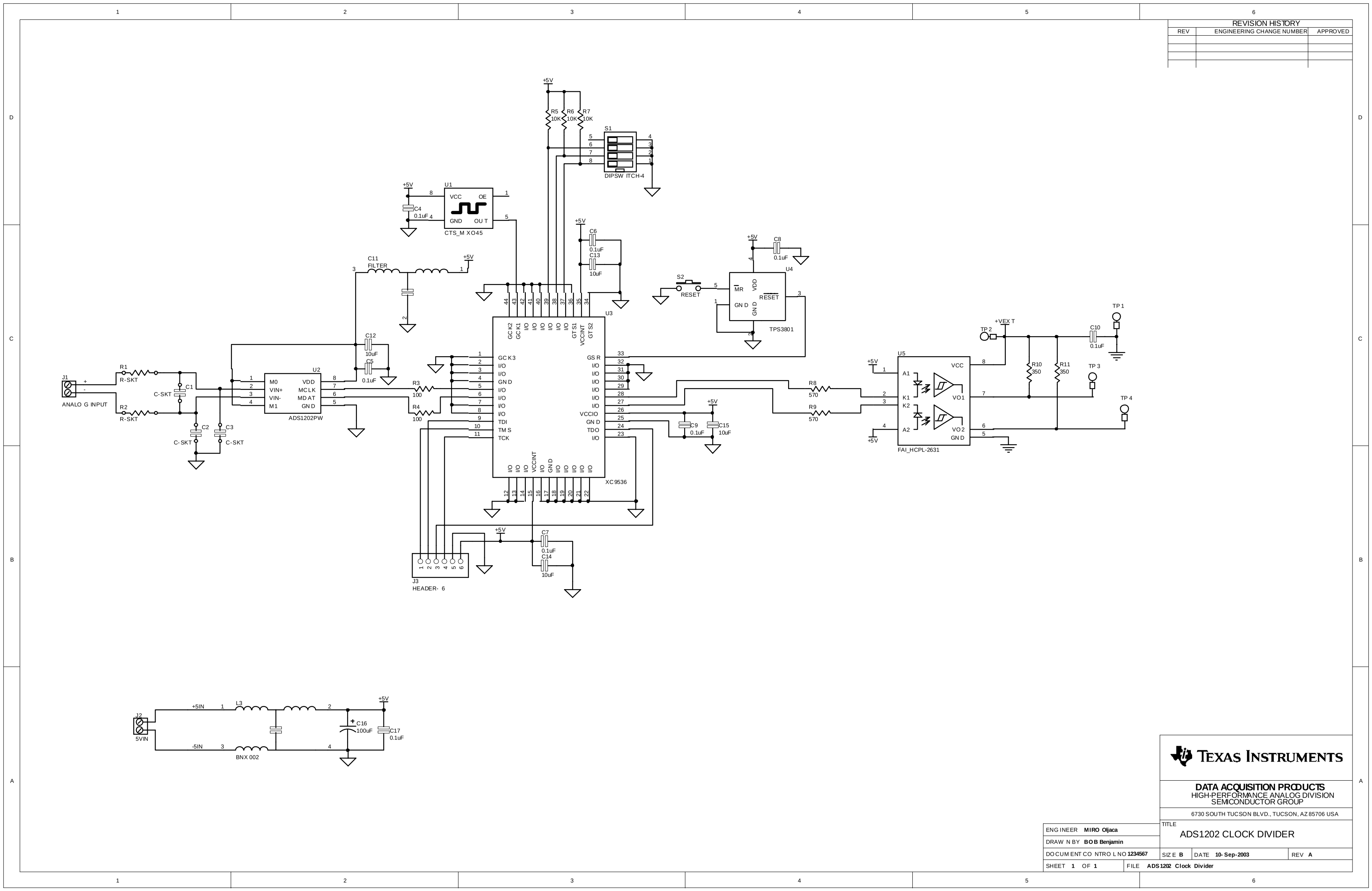
```
   process(counter_n, div_reg)
   begin
      cnt_en_n <= '0';
      case div_reg is
        when "000" => cnt_en_n <= '1';  -- divide by 1
        when "001" => cnt_en_n <= '1';  -- divide by 2
        when "010" =>
           if counter_n = "010" or counter_n = "001" then   -- divide by 3
              cnt_en_n <= '1';
           end if;
        when "011" => cnt_en_n <= '1';   -- divide by 4
        when "100" =>
           if counter_n = "100" or counter_n = "011" or
                 counter_n = "010" then   -- divide by 5
              cnt_en_n <= '1';
           end if;
        when "101" => cnt_en_n <= '1';   -- divide by 6
        when "110" =>
           if counter_n = "110" or counter_n = "101" or counter_n = "100" or
                 counter_n = "011" then   -- divide by 7
              cnt_en_n <= '1';
           end if;
        when "111" => cnt_en_n <= '1'; -- divide by 8
         when others => cnt_en_n <= '1';
      end case;
   end process;

   process(counter)
   begin
      counter_exp <= '0';
      if counter = "000" then
         counter_exp <= '1';
      end if;
   end process;

   process(CLK, resn)
   begin
      if resn = '0' then
         MCLK_int <= '0';
      elsif CLK'event and CLK = '1' then
         if counter_exp = '1' then
            MCLK_int <= (not MCLK_int);
         end if;
      end if;
   end process;

   MCLK_temp <= CLK when div_reg = "000" else (cnt_en_n and cnt_en);
   MCLK  <= MCLK_temp;
   MCLKn <= not MCLK_temp;

end RTL;
```

```
library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.std_logic_unsigned.all;
entity Clock_Data_IF is
   port (MCLK, resn  : in std_logic;
         MDAT        : in std_logic;
         MCLKm       : out std_logic;
         MDATm       : out std_logic
   );
end Clock_Data_IF;

architecture RTL of Clock_Data_IF is
   signal MDAT_change, MDAT_delay, MCLKm_int : std_logic;
   signal MCLK_CNT : std_logic_vector(1 downto 0);
begin
   process(resn, MCLK)
   begin
      if resn = '0' then
         MDAT_delay <= '0';
      elsif MCLK'event and MCLK = '1' then
         MDAT_delay <= MDAT;
      end if;
   end process;

   MDAT_change <= MDAT xor MDAT_delay;
         MCLK_CNT <= "00";
      elsif MCLK'event and MCLK = '1' then
         if MDAT_change = '1' then
            MCLK_CNT <= "00";
         else
            MCLK_CNT <= MCLK_CNT + 1;
         end if;
      end if;
   end process;

   process(resn, MCLK)
   begin
      if resn = '0' then
         MCLKm_int <= '0';
      elsif MCLK'event and MCLK = '1' then
         if MCLK_CNT  = "00" or MCLK_CNT = "10" then
            MCLKm_int <= not MCLKm_int;
         end if;
      end if;
   end process;

   MDATm <= MDAT_delay;
   MCLKm <= MCLKm_int;
end RTL;
```

```
library IEEE;
use IEEE.std_logic_1164.all;
entity CDI is
   port( CLK, resn  : in std_logic;
         M0, M1, M2 : in std_logic;
         MCLK       : out std_logic;
         MCLKm      : out std_logic;
         MDAT       : in std_logic;
         MDATm      : out std_logic
   );
end CDI;

architecture RTL of CDI is

   component Clock_Divider
      port (M0, M1, M2 : in std_logic;
            CLK, resn  : in std_logic;
            MCLK       : out std_logic;
            MCLKn      : out std_logic
      );
   end component;

   component Clock_Data_IF
      port (resn        : in std_logic;
            MCLK, MDAT  : in std_logic;
            MCLKm       : out std_logic;
            MDATm       : out std_logic
      );
   end component;

   signal MCLKn, MCLK_int : std_logic;
begin

   CLK_DIV_comp : Clock_Divider
      port map( M0    => M0,
                M1    => M1,
                M2    => M2,
                CLK   => CLK,
                resn  => resn,
                MCLK  => MCLK_int,
                MCLKn => MCLKn);

   CLK_DATA_comp : Clock_Data_IF
      port map( resn  => resn,
                MDAT  => MDAT,
                MCLKm => MCLKm,
                MDATm => MDATm,
                MCLK  => MCLK_int);

   MCLK <= MCLK_int;
end RTL;
```

APPENDIX B.

Schematic of implemented CDI circuit.



IMPORTANT NOTICE

Texas Instruments Incorporated and its subsidiaries (TI) reserve the right to make corrections, modifications, enhancements, improvements, and other changes to its products and services at any time and to discontinue any product or service without notice. Customers should obtain the latest relevant information before placing orders and should verify that such information is current and complete. All products are sold subject to TI's terms and conditions of sale supplied at the time of order acknowledgment.

TI warrants performance of its hardware products to the specifications applicable at the time of sale in accordance with TI's standard warranty. Testing and other quality control techniques are used to the extent TI deems necessary to support this warranty. Except where mandated by government requirements, testing of all parameters of each product is not necessarily performed.

TI assumes no liability for applications assistance or customer product design. Customers are responsible for their products and applications using TI components. To minimize the risks associated with customer products and applications, customers should provide adequate design and operating safeguards.

TI does not warrant or represent that any license, either express or implied, is granted under any TI patent right, copyright, mask work right, or other TI intellectual property right relating to any combination, machine, or process in which TI products or services are used. Information published by TI regarding third-party products or services does not constitute a license from TI to use such products or services or a warranty or endorsement thereof. Use of such information may require a license from a third party under the patents or other intellectual property of the third party, or a license from TI under the patents or other intellectual property of TI.

Reproduction of information in TI data books or data sheets is permissible only if reproduction is without alteration and is accompanied by all associated warranties, conditions, limitations, and notices. Reproduction of this information with alteration is an unfair and deceptive business practice. TI is not responsible or liable for such altered documentation.

Resale of TI products or services with statements different from or beyond the parameters stated by TI for that product or service voids all express and any implied warranties for the associated TI product or service and is an unfair and deceptive business practice. TI is not responsible or liable for any such statements.

Following are URLs where you can obtain information on other Texas Instruments products and application solutions:

Products		**Applications**	
Amplifiers	amplifier.ti.com	Audio	www.ti.com/audio
Data Converters	dataconverter.ti.com	Automotive	www.ti.com/automotive
DSP	dsp.ti.com	Broadband	www.ti.com/broadband
Interface	interface.ti.com	Digital Control	www.ti.com/digitalcontrol
Logic	logic.ti.com	Military	www.ti.com/military
Power Mgmt	power.ti.com	Optical Networking	www.ti.com/opticalnetwork
Microcontrollers	microcontroller.ti.com	Security	www.ti.com/security
		Telephony	www.ti.com/telephony
		Video & Imaging	www.ti.com/video
		Wireless	www.ti.com/wireless

Mailing Address: Texas Instruments
Post Office Box 655303 Dallas, Texas 75265