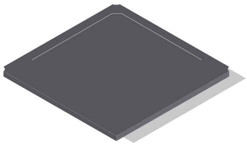


WIDEBAND QUAD DIGITAL DOWN-CONVERTER/UP-CONVERTER

FEATURES

- **Four Independently Configurable Wideband Down-Converter or Up-Converter Channels**
  - **Four Channel Down Convert Mode**
  - **Four Channel Up Convert Mode**
  - **Two Channels Down and Two Channels Up Mode**
- **Down-Conversion Channel Mode**
  - **Input Rates to 150-MSPS for Four Channels, 300-MSPS for Two Channels in Double Rate Mode**
  - **Four Wideband Down-Conversion Channels Support UMTS Standards**
  - **115-dB SFDR**
  - **FIR Filter Block Consists of 16 Cells That Provide Up to 256 Taps Per Channel**
  - **64 Parallel Input Bits and 64 Parallel Output Bits Provide Flexible I/O Options**
  - **Many Multiplex Output Options**
- **Up-Conversion Channel Mode**
  - **Output Rates to 150-MSPS for Four Channels, 300-MSPS for Two Channels**
  - **Four Up-Conversion Channels Support UMTS Standards**
  - **FIR Filter Block Consists of 16 Cells That Provide up to 256 Taps Per Channel**
  - **64 Parallel Input Bits and 64 Parallel Output Bits Provide Flexible I/O Options**
  - **Multiple Real and Complex Outputs**
  - **Two Channel Double Rate Real Output Mode With Rates to 300 MSPS**
  - **Outputs Can Be Independent, Summed Into Two or One Output(s), and Optionally Merged With Multiple GC5016 Chips**
- **JTAG Boundary Scan**
- **3.3-V I/O, 1.8-V Core**
- **Power Dissipation: <1 W for Four Channels**
- **Package: 252-Ball, 17-mm PBGA, 1-mm Pitch**

APPLICATIONS

- **Cellular Base Transceiver Station Transmit and Receive Channels**
  - **WCDMA**
  - **CDMA2000**
- **Radar**
- **General Filtering**
- **Test and Measurement**

Table of Contents

Please be aware that an important notice concerning availability, standard warranty, and use in critical applications of Texas Instruments semiconductor products and disclaimers thereto appears at the end of this data sheet.

DESCRIPTION

The GC5016 is a flexible wideband 4-channel digital up-converter and down-converter. The GC5016 is designed for high-speed, high bandwidth digital signal processing applications like 3G cellular base transceiver station transmit and receive channels. The GC5016 is also applicable for general-purpose digital filtering applications. The four identical processing channels can be independently configured for up-conversion, down-conversion, or a combination of two up-conversion and two down-conversion channels.

In up-conversion mode, the channel accepts real or complex signals, interpolates them by programmable amounts ranging from 1 to 4096, and modulates them up to selected center frequencies. The modulated signals are then summed together with other channels and optionally summed with modulated signals from other GC5016 chips. Channels can be used in pairs to increase the output sample rate, to increase filtering capacity, to increase the input bandwidth, or any combination. Each channel contains a user programmable input filter (PFIR), which can be used to shape the transmitted signal’s spectrum or can be used as a Nyquist transmit filter for shaping digital data such as QPSK, GMSK, or QAM symbols.

In down-conversion mode, the channel accepts real or complex signals, demodulates them from selected carrier frequencies, decimates them by programmable amounts ranging from 1 to 4096, applies a gain from a user defined automatic gain control, and produces 20-bit outputs. The frequencies and phase offsets of the four sine/cosine sequence generators can be independently specified, as can the interpolation and filtering of each circuit. Channels can be synchronized to support beam forming or frequency hopped systems. The output from the down-conversion channel is formatted and output in up to four output ports as either real or complex data.

ORDERING INFORMATION

PART NAME	TEMPERATURE	PACKAGE	DESCRIPTION
GC5016-PB	−40°C to 85°C	GDJ (S-PBGA-N252)	252 ball PBGA
GC5016-PBZ	−40°C to 85°C	ZDJ (S-PBGA-N2522)	252 ball lead free PBGA

These devices have limited built-in ESD protection. The leads should be shorted together or the device placed in conductive foam during storage or handling to prevent electrostatic damage to the MOS gates.

ABSOLUTE MAXIMUM RATINGS

over operating free-air temperature range unless otherwise noted[1]

		GC5016
Pad ring supply voltage, V_{PAD}		−0.3 V to 4 V
Core supply voltage, V_{CORE}		−0.3 V to 2.3 V
Input voltage (undershoot and overshoot), V_{IN}		−0.5 V to V_{PAD}+0.5 V
Storage temperature, T_{stg}		−65°C to 150°C
Junction temperature, T_J		105°C
Lead soldering temperature (10 seconds)		300°C
ESD classification	Human body model	2 kV
	Machine body model	200 V
	Charged device model	500 V
Moisture sensitivity		Level 3

[1] Stresses beyond those listed under “absolute maximum ratings” may cause permanent damage to the device. These are stress ratings only, and functional operation of the device at these or any other conditions beyond those indicated under “recommended operating conditions” is not implied. Exposure to absolute-maximum-rated conditions for extended periods may affect device reliability.

RECOMMENDED OPERATING CONDITIONS

	MIN	MAX	UNITS
Pad ring supply voltage, V_{PAD}	3	3.6	V
Core supply voltage, V_{CORE}	1.6	2	V
Temperature ambient, no air flow, T_A[1]	−40	85	°C
Junction temperature, T_J[2]		105	°C

[1] Chips specifications in the *AC CHARACTERISTICS* and *DC CHARACTERISTICS* tables are production tested at 100°C case temperature. QA tests are performed at 85°C case temperature.

[2] Thermal management may be required for full rate operation, see the *THERMAL CHARACTERISTICS* table . The circuit is designed for junction temperatures up to 125°C. Sustained operation at elevated temperatures reduces long-term reliability. Lifetime calculations based on maximum junction temperature of 105°C.

DC CHARACTERISTICS

−40°C to 85°C case temperature unless otherwise noted

PARAMETER		V_{PAD} = 3 V to 3.6 V			UNIT
		MIN	TYP	MAX	
V_{IL}	Voltage input low [1]			0.8	V
V_{IH}	Voltage input high [1]	2			V
V_{OL}	Voltage output low (I_{OL} = 2 mA) [1]			0.5	V
V_{OH}	Voltage output high (I_{OH} = −2 mA) [1]	2.4			V
$\| I_{IN} \|$	Leakage current (V_{IN} = 0 V or VPAD), inputs or outputs in tristate condition [1]			1	µA
| IPU |	Pullup current (V_{IN} = 0 V) (TDI, TMS, TCK) [1]	5		35	µA
I_{CCQ}	Quiescent supply current, I_{CORE} or I_{PAD}(V_{IN}=0 or V_{PAD}, $\overline{RST}$ = $\overline{TRST}$ = 0) [1]			4	mA
C_{IN}	Data input capacitance (all inputs except CK) [2]		4		pF
C_{CK}	Clock input capacitance (CK input) [2]		13		pF

[1] Each part is tested with a 100°C case temperature for the given specification.

[2] Controlled by design and process and not directly tested.

NOTE: General: Voltages are measured at low speed. Output voltages are measured with the indicated current load.
General: Currents are measured at nominal voltages, high temperature (100°C for production test, 85°C for QA).

AC CHARACTERISTICS

−40°C to 85°C case, supplies across recommended range unless otherwise noted

PARAMETER		MIN	MAX	UNITS
f_{CK}	Clock frequency [1]	[3]	160	MHz
t_{CKL}	Clock low period (below V_{IL}) [1]	2		ns
t_{CKH}	Clock high period (above V_{IH}) [1]	2		ns
t_r, t_f	Clock rise and fall times (V_{IL} to V_{IH}) [4]		2	ns
t_{su}	Input set up before CK goes high (AI, BI, CI, DI, SIA, or SIB) [1]	2		ns
t_h	Input hold time after CK goes high [1]	0.5		ns
t_d	Data output delay from rising edge of CK. (AO, BO, CO, DO, IFLG, [A−D]FS, [A−D]CK, or SO) [1]		5	ns
$t_{h(O)}$	Data output hold from rising edge of CK [1]	1		ns
f_{JCK}	JTAG clock frequency [1]		40	MHz
t_{JCKL}	JTAG clock low period (below V_{IL}) [1]	10		ns
t_{JCKH}	JTAG clock high period (above V_{IH}) [1]	10		ns
$t_{su(J)}$	JTAG input (TDI or TMS) set up before TCK goes high [1]	1		ns
$t_{h(J)}$	JTAG input (TDI or TMS) hold time after TCK goes high [1]	10		ns
$t_{d(J)}$	JTAG output (TDO) delay from falling edge of TCK [1]		10	ns
$t_{su(C)}$	Control setup during reads or writes. [1][5]	2		ns
$t_{su(EWC)}$	Control data setup during writes (normal mode). [1][5]	4		ns
$t_{h(C)}$	Control hold during writes. [1][5]	1		ns
t_{CSPW}	Control strobe (CE and WR low) pulse width (write operation). [1][5]	20		ns
$t_{d(C)}$	Control output delay CE and RD low and A stable to C (read operation). [1][5]		12	ns
t_{REC}	Control recovery time between reads or writes. [1][5]		20	ns
$t_{(CZ)}$	End of read to HI-Z [2][5]		5	ns
$I_{C(DYN)}$	Core dynamic supply current nominal voltages, 100 MHz, four channels active, full length filters, high temperature. [6]		420	mA

[1] Each part is tested with a 100°C case temperature for the given specification. Lots are sample tested at −40°C.
[2] Controlled by design and process and not directly tested.
[3] The minimum clock rate is calculated in the cmd5016 configuration program. It may be estimated by (1 + *ncic − nfir)* x 200 kHz.
[4] Recommended practice
[5] See Figure 27 through Figure 30.
[6] Each port is tested with a 100°C case temperature for the given specification.
General: Timing is measured from CK at VPAD/2 to input or output at VPAD/2. Output loading is a 50-Ω transmission line whose delay is calibrated out.

THERMAL CHARACTERISTICS

THERMAL CONDUCTIVITY	252 BGA	UNITS
	1 W	
Theta junction to ambient, θ_{JA}	22	°C/W
Theta junction to case, θ_{JC}	5	°C/W

NOTE: Air flow reduces θ_{JA} and is highly recommended.

POWER CONSUMPTION

The maximum power consumption is a function of the operating mode of the chip. The following equation estimates the typical power supply current for the chip. Chip-to-chip variation is typically ±5%. The *AC Characteristics* provides the production test limit for current in a maximum configuration. It is 10% over the typical value.

$$\text{Icore} = (f_{CK}/100\text{ MHz})\ (\text{Vcore}/1.8\text{ V})\ (\mathit{Number\_of\_Active\_Channels}/4)\ (0.75 + \text{FIRDutyCycle})\ 220\text{ mA}$$

The FIRDutyCycle is calculated in the cmd5016 configuration software described later. It can be estimated by:

Down Converter Mode:

$$\text{FIRDutyCycle} = 1 \text{ for } f_{CK}/\text{Fout} \le 16$$

$$16 \times \text{Fout}/f_{CK} \text{ otherwise}$$

Up Converter Mode:

FIRDutyCycle = 1 for f_{CK}/Fin ≤ 32
32 x Fin/f_{CK} otherwise

Current consumption on the pad supply is primarily due to the external loads and follows C x V x F. Internal loads are estimated at 2 pF per pin. Data outputs have a transition density of going from a zero to a one once per four clocks, while clock outputs transition every cycle. The frame strobes consume negligible power due to the low transition frequency. In general,

$$I_{pad} = \Sigma\, DataPad/4 \times C \times F \times V + \Sigma\, ClockPad \times C \times F \times V$$

Typically loads are 20 pF per pin. A worst case current would be all four output ports operating at 125 MHz and the four output clocks with [A−D]CK active at 125 MHz.

$$I_{pad} = (64/4 + 4) \times (C+2pF) \times Fout \times Vpad = 20 \times 22\ pF \times 125\ MHz \times 3.3\ V = 180\ mA$$

FUNCTIONAL BLOCK DIAGRAM

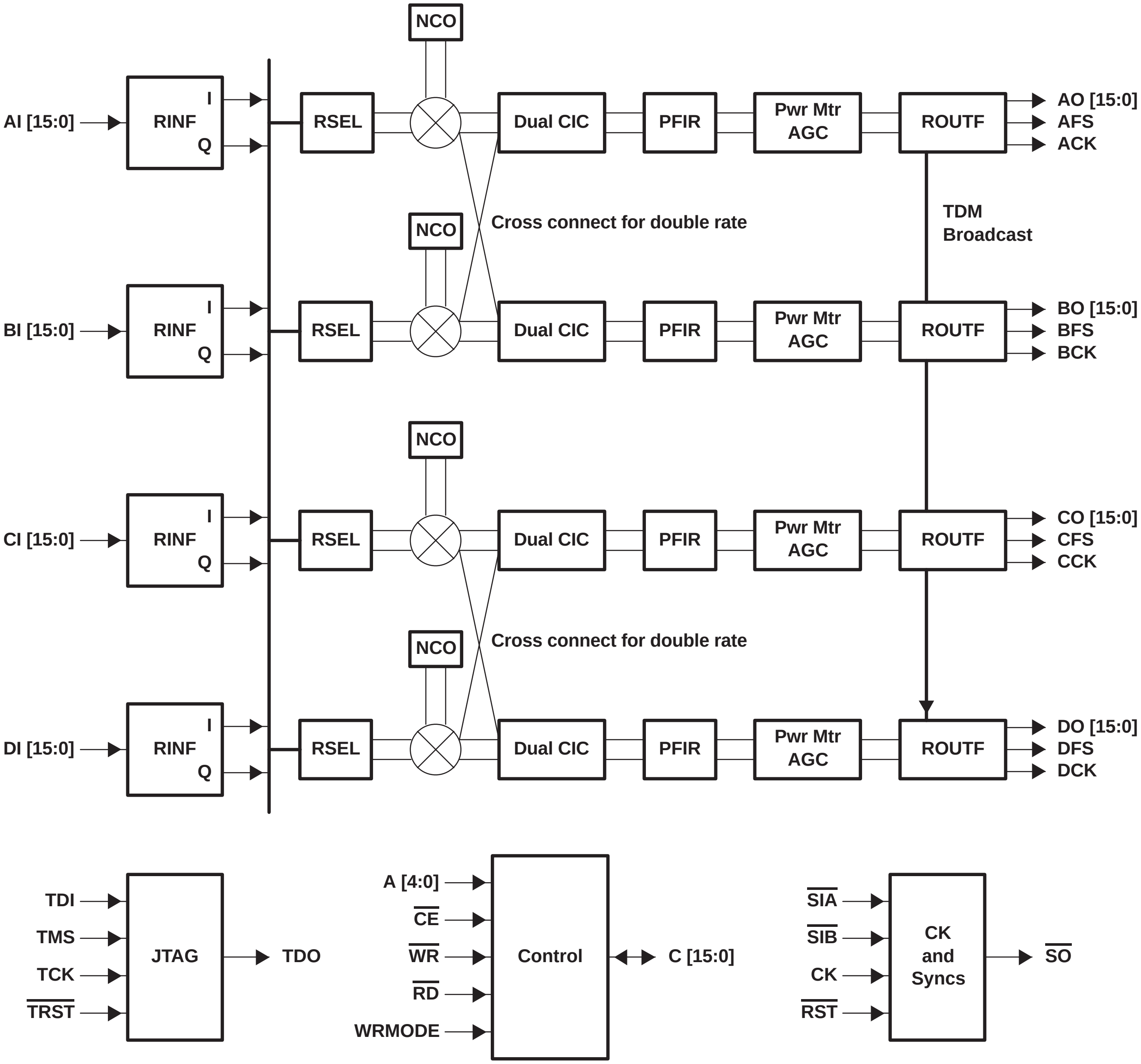


Figure 1. GC5016 in Digital Down-Conversion Mode

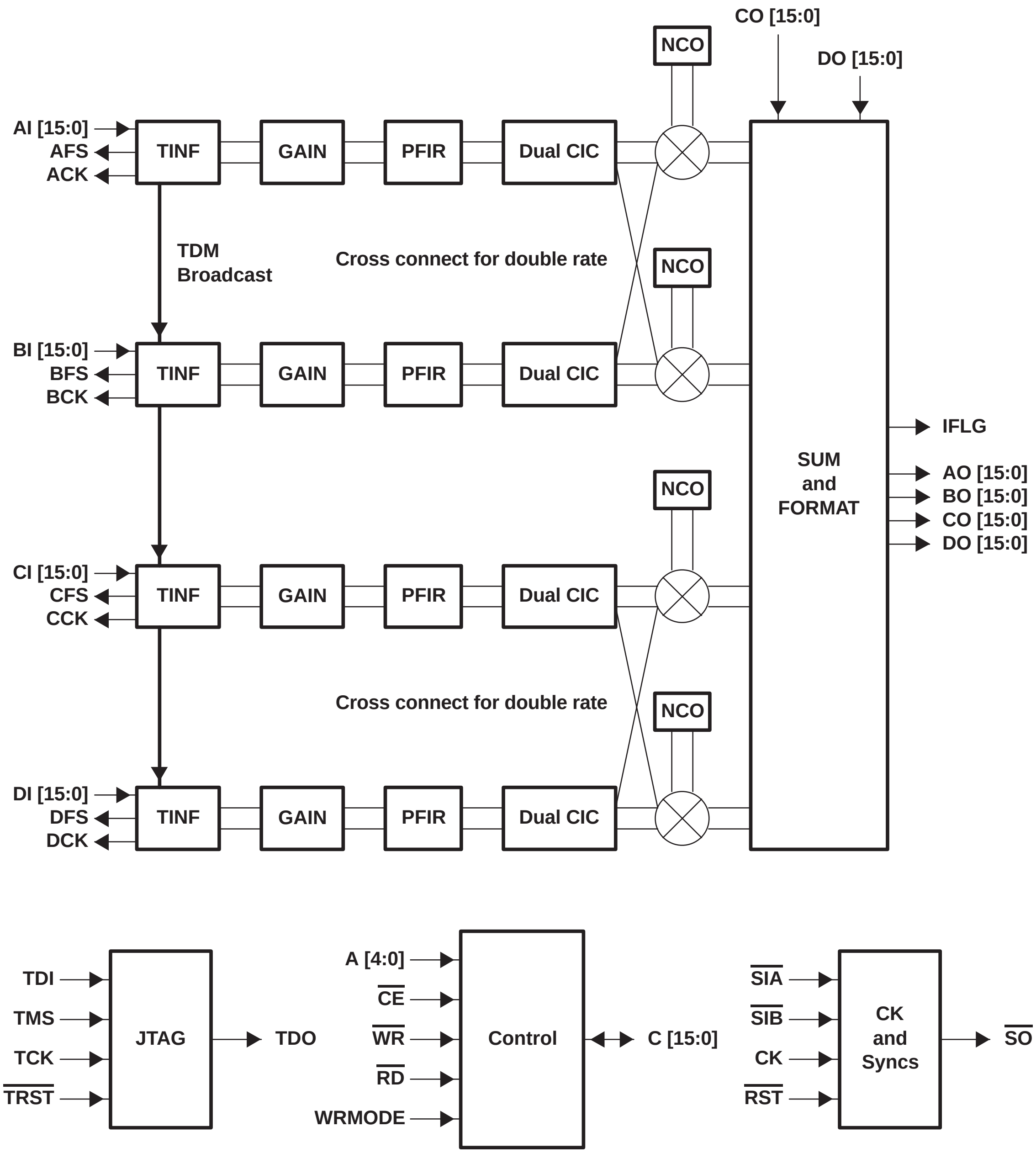


Figure 2. GC5016 in Digital Up-Conversion Mode

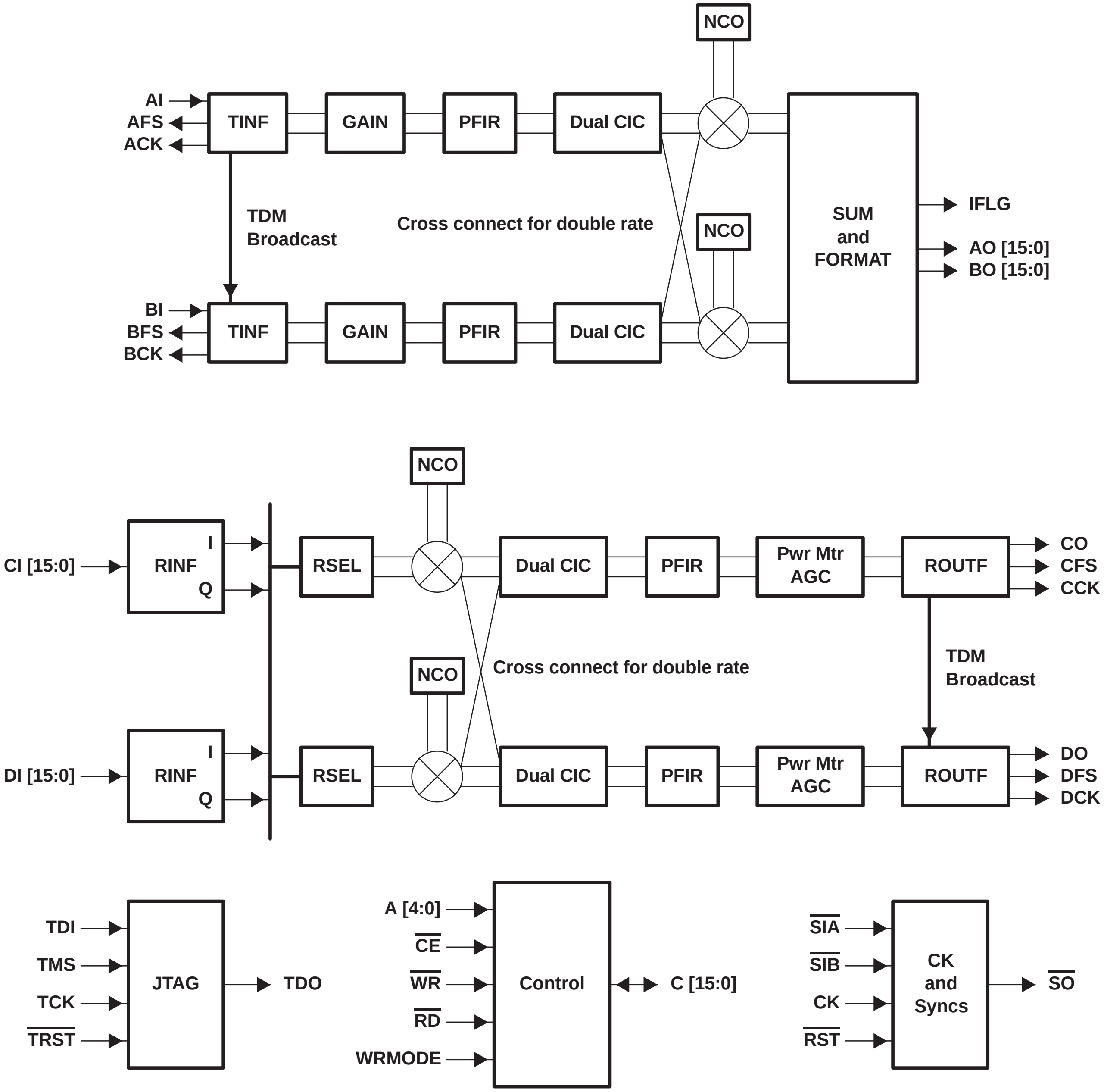


Figure 3. GC5016 in Transceiver Mode

PIN ASSIGNMENTS

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
A	GND	BCK	BO14	BO13	AO11	AO10	BO8	GND	GND	AO6	BO4	BO3	AO1	IFLG	TMS	GND
B	AI1	BI0	AO15	AO14	BO11	BO10	AO9	VPAD	VPAD	BO5	AO4	BO2	BO0	TDI	$\overline{\text{TRST}}$	$\overline{\text{RST}}$
C	BI3	BI2	AI0	AFS	BO15	AO12	BO9	AO7	BO6	AO5	AO3	BO1	$\overline{\text{SO}}$	GND	$\overline{\text{SIB}}$	CI15
D	AI5	AI3	AI2	BFS	ACK	AO13	BO12	BO7	AO8	AO2	AO0	TDO	TCK	DI15	DI14	CI13
E	BI6	AI6	BI5	BI1	GND	VPAD	VPAD	VPAD	VPAD	VPAD	VPAD	GND	$\overline{\text{SIA}}$	DI13	DI12	DI11
F	AI8	BI7	AI7	BI4	VCOR	GND	GND	VPAD	VPAD	GND	GND	VCOR	CI14	CI12	CI11	DI10
G	BI9	AI9	BI8	AI4	VCOR	GND	GND	GND	GND	GND	GND	VCOR	CI10	CI9	DI8	CI8
H	GND	AI10	BI11	BI10	VCOR	VCOR	GND	/	/	GND	VCOR	VCOR	DI9	VPAD	DI7	GND
J	GND	BI12	AI12	AI11	VCOR	VCOR	GND	/	/	GND	VCOR	VCOR	DI5	CI7	DI6	GND
K	AI13	BI13	AI14	BI14	VCOR	GND	GND	GND	GND	GND	GND	VCOR	CI5	DI4	CI4	CI6
L	AI15	BI15	CK	A0	VCOR	GND	GND	VPAD	VPAD	GND	GND	VCOR	CCK	DI2	CI3	DI3
M	A1	A2	A3	A4	GND	VPAD	VPAD	VPAD	VPAD	VPAD	VPAD	GND	DFS	CI1	DI1	CI2
N	$\overline{\text{CE}}$	$\overline{\text{RD}}$	WRMODE	C0	C3	C9	C10	DO1	CO2	CO7	CO12	DO13	CO15	DCK	CI0	DI0
P	$\overline{\text{WR}}$	C1	GND	C5	C8	C14	DO2	CO4	DO4	CO6	DO8	CO11	CO13	GND	DO15	CFS
R	C2	C4	C6	C12	C13	DO0	CO3	VPAD	VPAD	DO5	DO7	CO9	DO10	DO12	CO14	DO14
T	GND	C7	C11	C15	CO0	CO1	DO3	GND	GND	CO5	DO6	CO8	DO9	CO10	DO11	GND

/ = No Ball

Terminal Functions

Bit 0 is the least significant bit on all buses. All outputs are tri-statable. Jtag related inputs have pullups—if an external pulldown is used, it must be less than 500 Ω. Whenever I and Q are multiplexed, I comes first. All clocked inputs are registered on the rising edge of CK and all clocked outputs are released on the rising edge of CK, except for Jtag output (TDO).

SIGNAL	TYPE	DESCRIPTION
CONTROL I/O		
A[4..0]	I	Control address bus – Active high inputs These pins are used to address the control registers within the chip. Each of the control registers within the chip are assigned a unique address. A control register can be written to or read from by having the page register set to the appropriate page and then setting A[4..0] to the register’s address.
C[15..0]	I/O	Control data I/O bus – Active high bidirectional I/O pins This is the 16-bit control data I/O bus. Control registers are written to or read from through these pins. The chip drives these pins when $\overline{\text{CE}}$ is low, $\overline{\text{RD}}$ is low, and $\overline{\text{WR}}$ is high.
$\overline{\text{CE}}$	I	Chip enable – Active low input pin This control strobe enables the read or write operations.
$\overline{\text{WR}}$	I	Write enable – Active low input pin The value on the C[15..0] pins is written into the register selected by the A[4..0] and page register when $\overline{\text{WR}}$ and $\overline{\text{CE}}$ are low.
$\overline{\text{RD}}$	I	Read enable – Active low input pin The register selected by A[4..0] and the page register is output on the C[15..0] pins when $\overline{\text{RD}}$ and $\overline{\text{CE}}$ are low.
DATA I/O		
AI[15..0]	I	Clocked input port A, data bits 0 through 15 Can be configured for many possible input formats.
BI[15..0]	I	Clocked input port B, data bits 0 through 15 Can be configured for many possible input formats.
CI[15..0]	I	Clocked input port C, data bits 0 through 15 Can be configured for many possible input formats.
DI[15..0]	I	Clocked input port D, data bits 0 through 15 Can be configured for many possible input formats.
AO[15..0]	O	Clocked output port A, data bits 0 through 15 Can be configured for many possible output formats.
BO[15..0]	O	Clocked output port B, data bits 0 through 15 Can be configured for many possible output formats.
CO[15..0]	I/O	Dual function: Clocked output − port C, data bits 0 through 15 Can be configured for many possible output formats. Clocked input – Sum IO input data, data bits 0 through 15 Can be configured for many possible input formats.
DO[15..0]	I/O	Dual function: Clocked output − port D, data bits 0 through 15 Can be configured for many possible output formats. Clocked input – sum IO input data, data bits 0 through 15 Can be configured for many possible input formats.
[A..D]CK	O	Clocked output for ports [A..D] The clock for input ports in up-conversion mode and output ports in down-conversion mode. When configured as a transceiver, channels A and B are in up-conversion and channels C and D are in down-conversion mode.
[A..D]FS	O	Clocked output frame strobes for channels A..D Used to signify the beginning of a data frame for each input port in up-conversion mode and output in down-conversion mode. The frame strobes are set high by the GC5016 with the first word in a frame. The frame strobes can be programmed to be sent early.
CK	I	Main input clock. The clock input to the chip.

SIGNAL	TYPE	DESCRIPTION
DATA I/O (CONTINUED)		
IFLG	O	Clocked output A flag used to indicate which samples are real or imaginary in up-conversion mode when I and Q are time multiplexed.
WRMODE	I	A static control input that changes the timing of control writes. Normally tied low. When low control write data must be stable for a setup time ahead and hold time after the end of the write strobe. When high data must be stable for a setup time ahead of the write strobe going active until a hold time after it goes inactive.
$\overline{\text{RST}}$	I	Chip reset bar. Active low signal. Not clocked
$\overline{\text{SIA}}$	I	Sync input A bar. Active low data input signal
$\overline{\text{SIB}}$	I	Sync input B bar. Active low data input signal
$\overline{\text{SO}}$	O	Sync output bar. Active low data output signal
JTAG I/O		
TCK	I	JTAG clock – Active high input. Internal pullup
TDI	I	JTAG data in – Active high input clocked on TCK rising. Internal pullup
TDO	O	JTAG data out – Tristate output clocked on falling edge of TCK.
TMS	I	JTAG interface – Active high input clocked on TCK rising. Internal pullup
$\overline{\text{TRST}}$	I	Asynchronous JTAG reset bar. Internal pullup
SUPPLIES		
GND		Ground
VCOR		Core supply voltage. Used to supply the core logic, nominally set to 1.8 V.
VPAD		Interface voltage. Used to set the I/O levels for all pins, nominally set at 3.3 V.

GC5016 DOWN-CONVERSION MODE

Overview

Figure 1 shows the functional block diagram for the GC5016 when configured as a 4-channel digital down-converter. In a common configuration, each down-conversion channel is used to process a separate complex data with I and Q multiplexed. The input ports AI[15..0], BI[15..0], CI[15..0], and DI[15..0] accept up to 16-bit parallel data, enabling high input bandwidths. The receive input formatter allows data to be input in a variety of formats and multiple input ports, sending the real or complex, 16-bit data onto a bus distributed to all down-conversion channels. Each down-conversion channel selects an input signal from the bus, demodulates it from selected carrier frequency, decimates it using a 5-stage CIC filter by programmable amounts ranging from 1 to 256, decimates 1 to 16 using a user programmable FIR filter, measures power, applies either a fixed gain or a gain from a user defined automatic gain control, and produces up to 20-bit outputs. The frequency, phase offset, filter coefficients, power meter parameters, AGC parameters, and output can be independently specified for each of the four channels. Channels can be synchronized to support beam forming or frequency hopped systems. Two channels can be operated in tandem to allow double input bandwidth, double output bandwidth, or both.

Receive Input Formatter

The GC5016 has four 16-bit input ports AI[15..0], BI[15..0], CI[15..0], and DI[15..0] that allow for multiple input options in the down-conversion mode.

Receive Input Data Formats

Five data formats are supported:

- Up to four real sources at CK rate
- Up to four complex sources at CK/2 rate with I/Q time multiplexed
- Up to two complex sources at CK rate with I/Q on separate ports
- Up to two real sources at twice CK rate with even and odd time samples on different ports
- One complex source at twice CK rate with even and odd time samples on different ports

Each input port has a receive input data formatter attached to it. The data formatter accepts 16 bits from its input port and outputs a 16-bit I bus and a 16-bit Q bus (the rinf bus). When there is no data to send the output bus is held to zero. For example, if the input data is real the Q bus is zero. If the input data is complex, with the I/Q time multiplexed every other time sample is zero. If the input data is complex at 1x, the I data is expected in port A (or C) and A's Q bus is zero's, while the imaginary data is expected in port B (or D) and B's I bus is zero.

The input format can be specified to the cmd5016 software by setting *rin_rate* and *rin_cmplx*. These are user variables in the software that do not directly correspond to hardware registers. Alternately, the user may directly program the control register bit fields (in this case, *rinf_sel_A* through *rinf_sel_D*, *mix_rcv_sel*, and *mix_rcv_cmplx*). In general, it is encouraged for users to use the software as this provides an easier user interface and error checking. Table 1 shows the modes, the inputs, programming using software, and programming directly in hardware. For example, for the mode with four complex inputs, data from source 1 is entered time multiplexed I, followed by Q onto port AI. Configuration using the software requires that *rin_cmplx* be set to 1 and *rin_rate* be set to 0 (half rate). Alternatively, if the user wishes to directly program the hardware register fields, *rinf_sel_A* should be set to 3, *mix_rcv_sel* to 0 for channel A, and *mix_rcv_cmplx* to 0 for channel A (etc., for channels B, C, and D).

NOTE: Names in italics refer to parameter inputs to the cmd5016 software configuration program.

Table 1. Receive Input Modes and Controls

	INPUT PORTS				SOFTWARE CONTROLS	FIELDS FOR CHANNELS A, B, C, AND D *rinf_sel / mix_rcv_sel / mix_rcv_cmplx*			
MODE	**AI**	**BI**	**CI**	**DI**	*rin_cmplx / rin_rate*	**A**	**B**	**C**	**D**
Four real	1I	2I	3I	4I	0/1	4/0/0	4/1/0	4/2/0	4/3/0
Four complex	1I/1Q	2I/2Q	3I/3Q	4I/4Q	1/0	3/0/0	3/1/0	3/2/0	3/3/0
Two complex	1I	1Q	2I	2Q	1/1	4/0/1	1/x/x	4/2/1	1/x/x
Two double rate real	1I(2k)	1I(2k+1)	2I(2k)	2I(2k+1)	0/2	4/0/0	4/1/0	4/2/0	4/3/0
One double rate complex	I(2k)	Q(2k)	I(2k+1)	Q(2k+1)	1/2	4/0/1	1/x/1	4/2/x	1/x/x

Synchronization for IQ Multiplexed Mode

When I and Q are time multiplexed, a synchronization signal is used to determine which sample is I and which is Q. The input data is delayed by one cycle to form the I stream and is directly output for the Q stream. Thus far the data stream is (I0,Q0), (Q0, I1), (I1, Q1), … where I0 is the real portion of the sample at time 0. Then every other complex sample is zeroed using receive interpolation discussed below, so that the stream is now (I0,Q0), (0,0), (I1,Q1), (0,0), … . The timing for sync relative to the data sample is shown in the next section.

Receive Interpolation

At times it is desired to operate the chip at a clock that is a multiple of the sample rate. This situation occurs when the sample rate is relatively low, but the processing requirements are high. In this case, the chip can be programmed to insert 0−15 zeros (*rinf_zpad*) between input samples. This effectively interpolates the signal up by *rinf_zpad*+1. The higher CK rate means the chip is operating faster so the PFIR has more multiplies per second available. It also allows greater flexibility in selecting the output sample rate since *Fs_out*=Fad x (1+*rinf_zpa*d) / (*cic_dec* x *fir_dec*).

One sample is registered while the data input on the other *rinf_zpad* clocks are zeroed. The user has control over which sample is used by using *rinf_zpad_sync*. The sync encounters a two CK cycle delay, then loads a counter. When the counter reaches the terminal count, the counter is reloaded and a data sample is kept. All other data samples are zeroed. The sample two cycles after the sync is used, while the other samples are ignored. The sync input may be periodic in any multiple of (*rinf_zpad*+1) or may occur just once.

If I and Q are time multiplexed, then the sync should be two clocks before the Q sample, since the I sample is delayed by one to align it with the Q. If I and Q are time multiplexed, the minimum receive interpolation is two (since we must have that much time just to get the data in). Larger interpolations are also possible as shown in Figure 4.

For real inputs, rinf_zpad can do receive interpolation as well. The same timing applies as is the IQ multiplex case (shown in Figure 4), except the real inputs come in at the Q word time and the I word time would be a don't care.

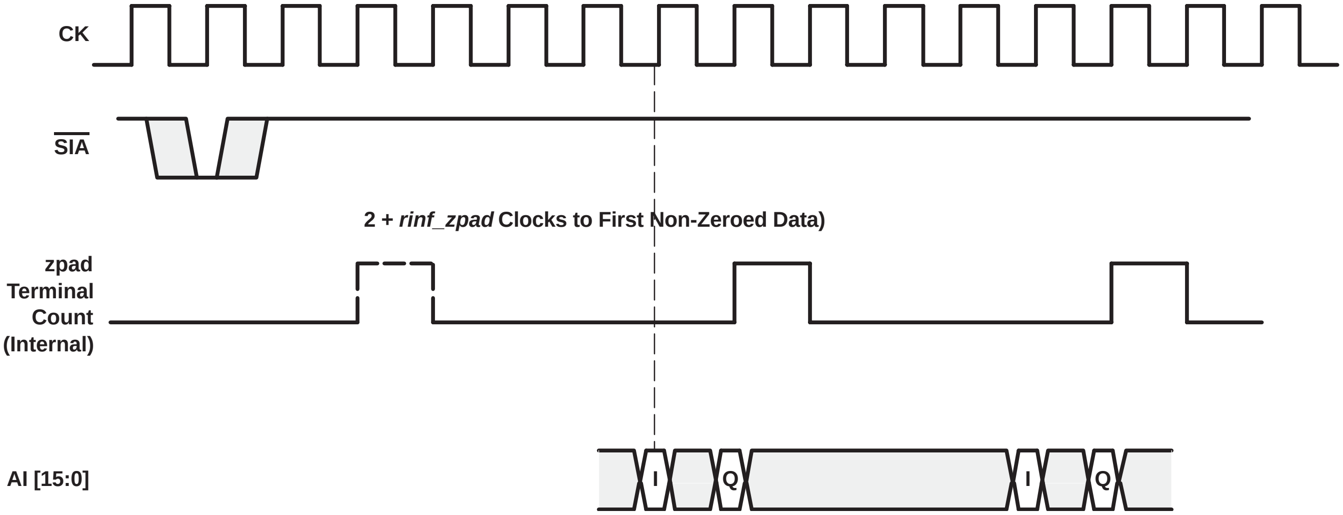

CK
SIA
2 + rinf_zpad Clocks to First Non-Zeroed Data)
zpad Terminal Count (Internal)
AI [15:0]
I
Q
I
Q

Figure 4. IQ Multiplexed With zpad = 4

Receiver Desensitizing

In a few circumstances, there is a need to reduce the receiver sensitivity, which can be done by adding noise to the signal. The GC5016 allows this to be done digitally by adding pseudo random noise to the input data stream. The noise power is added by bit wise xoring the input data stream with a PN sequence. The user has control over the noise power by programming which bits get the noise added. The noise power can go from –3 dBFS (0xffff) to −99 dbFS (0x1). This is programmed using *rcv_noise_A* (or B, C, or D). The noise uses the PN generator that is also used for diagnostics. The generator must be enabled for this feature to work. See the Diagnostics section.

Receive Input Selection

In each channel an input selector exists at the input to the mixer. This selects I and Q data from one of four receive input formatters. The field *mix_rcv_sel* allows selection of the rinf bus. In the event of a complex input at CK rate (or twice CK), the I data is on rinf bus A, while the corresponding Q data is on rinf bus B. Setting *mix_rcv_cmplx* allows the mixer to select I from A and Q from B. The software automatically calculates *mix_rcv_cmplx*.

Mixer

Each of the four multipliers (I x cos, I x sin, Q x cos, Q x sin) can be programmed in one of four modes (off, receive, cross transmit, normal transmit) and can be inverted (multiplied by –1). When down converting each multiplier should be programmed to be off or active in receive mode depending on whether there is nonzero input data for the multiplier. Programming Q x sin to be inverted corresponds to a mathematical view of down-conversion (mix with negative frequency tone to get a positive spectrum). Programming I x sin to be inverted corresponds to a radio view (tune to a frequency to get the signal at that frequency). The fields involved are *mix_icos*, *mix_isin*, *mix_qco*s, *mix_qsin*, and *mix_inv_icos*, *mix_inv_isin*, *mix_inv_qcos*, and *mix_inv_qsin*. The software automatically programs these fields for you assuming a mathematical view.

Data is accepted into the mixer as 16-bit data, placed into the upper bits of an 18-bit word, multiplied by a 20-bit NCO word, summed with the output of a second multiplier creating a 21-bit output, which is sent to the CIC. This means there is a 6-dB attenuation going through the mixer, in other words, there is a 1-bit growth on top to allow for the extreme case of both real and imaginary inputs at full scale being multiplied by an NCO word that is at 45 degrees. For real inputs, the attenuation is 6 dB, so the CIC can safely be programmed to have 6-dB gain. For complex inputs, the attenuation is 3-dB peak. This has implications on programming the gain throughout the signal path that is addressed in a later section. Figure 5 shows the multiplexing options in the mixer.

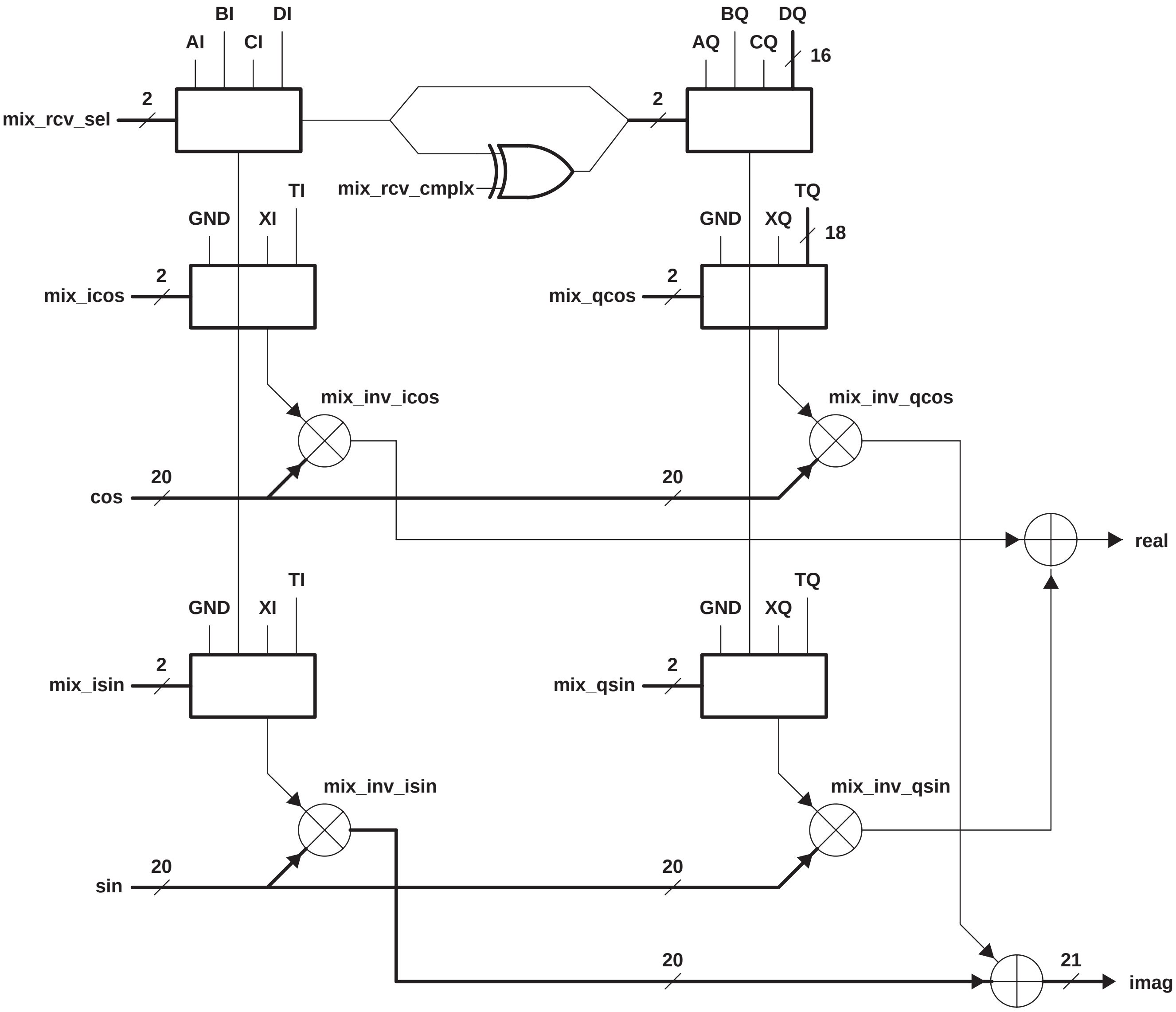


Figure 5. Multiplexing Options in Mixer

Numerically Controlled Oscillator (NCO)

The tuning frequency of each up-converter is specified as a 48-bit word and the phase offset is specified as a 16-bit word. The NCOs can be synchronized with NCOs on other chips. This allows multiple down converter outputs to be coherently combined, each with a unique phase and amplitude. A block diagram of the NCO circuit is shown in Figure 6. The tuning frequency is set to FREQ according to the formula FREQ = (2^{48}) x F/f_{CK}, where F is the desired tuning frequency and f_{CK} is the chip's clock rate. The 16-bit phase offset setting is *phase* = (2^{16}) x Ph/2π, where Ph is the desired phase in radians ranging between 0 and 2π. Note that a negative tuning frequency should be used for down-conversion. A positive tuning frequency can be used to spectrally flip the spectrum of the desired signal (if the input is real). FREQ and *phase* are set as shown in Table 53 through Table 56 or in software by specifying *freq_msb*, *freq_mid*, *freq_lsb*, and *phase*. The configuration software calculates the appropriate settings for *freq_msb*, *freq_mid*, and *freq_lsb* given the chip clock frequency (*f_{CK}*) and *freq*. (If both are set *freq_msb* takes priority). The calculation includes the effects of zpad and double rate processing. Both *fck* and *freq* should be stated in MHz.

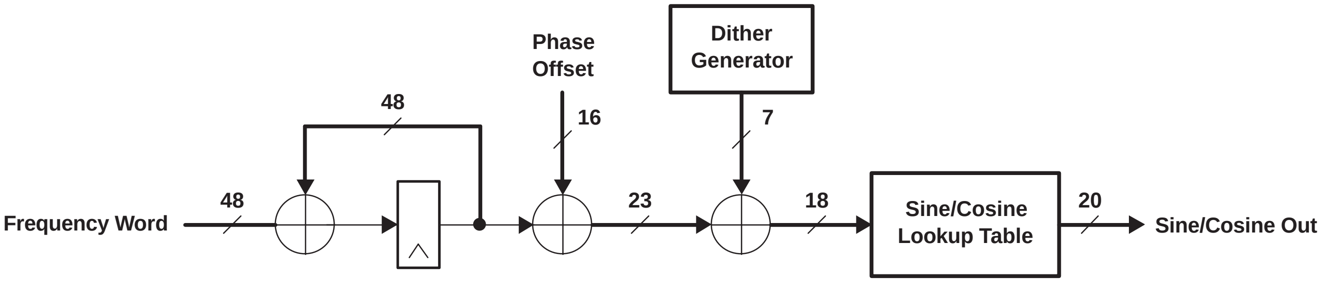
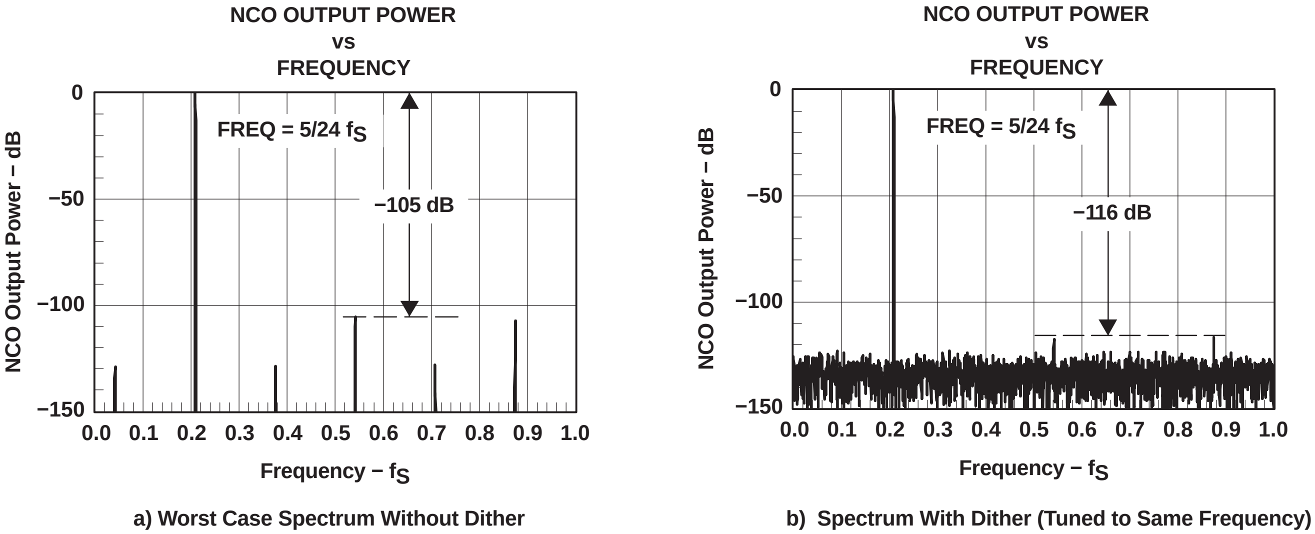


Figure 6. Numerically Control Oscillator (NCO) Circuit

The NCO’s frequency, phase and accumulator can be initialized and synchronized with other channels using the *freq_sync*, *phase_sync*, and *nco_sync* controls. The *freq_sync* and *phase_sync* controls determine when new frequency and phase settings become active. Normally, these are set to always so that they take effect immediately, but can be used to synchronize frequency hopping or beam forming systems. The *nco_sync* control is usually set to never, but can be used to synchronize the NCOs of multiple channels.

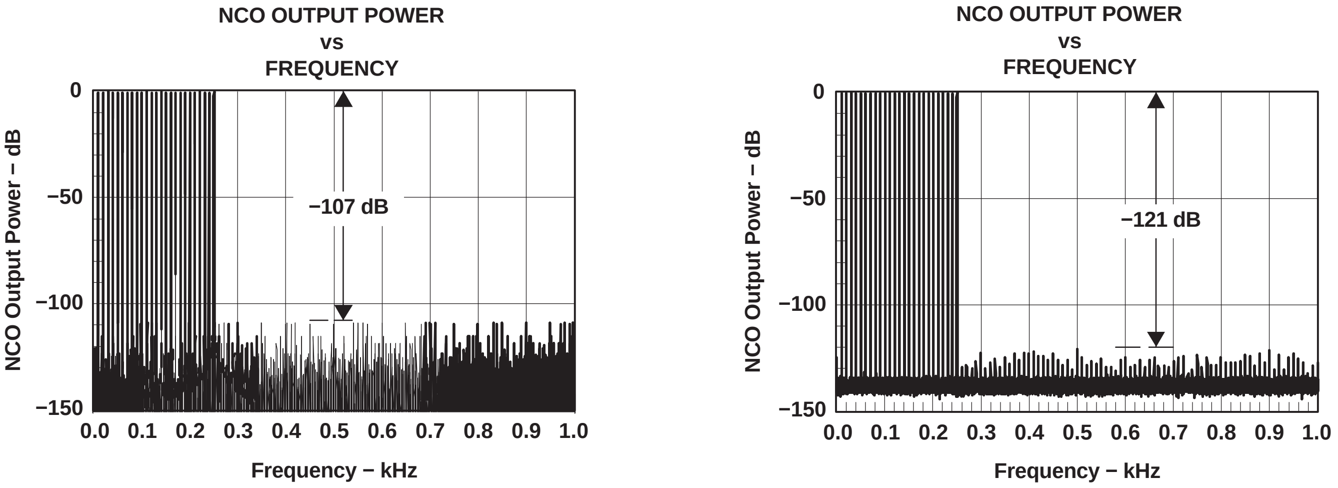
The NCO’s spur level is reduced to below −113 dB through the use of phase dithering. The spectrums in Figure 7 show the NCO spurs for a worst case tuning frequency with and without dithering. Dithering decreases the spur level from −105 dB to −116 dB. Dithering is turned on or off using the *dither_sync* controls. Holding *dither_sync* always on freezes the dither value, effectively turning off dither.



a) Worst Case Spectrum Without Dither

b) Spectrum With Dither (Tuned to Same Frequency)

Figure 7. Example NCO Spurs With and Without Dithering



a) Plot Without Dither or Phase Initialization

b) Plot With Dither and Phase Initialization

Figure 8. NCO Peak Spur Plot

The worst-case NCO spurs at −113 dB to −116 dB, such as the one shown in Figure 7(b), are due to a few frequencies that are related to the sampling frequency by multiples of $f_{CK}/96$ and $f_{CK}/124$. In these cases, the rounding errors in the sine/cosine lookup table repeat in a regular fashion, thereby concentrating the error power into a single frequency, rather than spreading it across the spectrum. These worst-case spurs can be eliminated by selecting an initial phase that minimizes the errors or by changing the tuning frequency by a small amount (50 Hz). Setting the initial phase to 4 for multiples of $f_{CK}/96$ or $f_{CK}/124$ (and to 0 for other frequencies) results in spurs below −115 for all frequencies.

Figure 8 shows the maximum spur levels as the tuning frequency is scanned over a portion of the frequency range with the peak hold function of the spectrum analyzer turned on. Notice that the peak spur level is −107 dB before dithering and is −121 dB after dithering has been turned on and the phase initialization described above has been used.

Double rate processing is done by sending time samples (2k) to mixer A and time samples (2k+1) to mixer B. The frequency is tuned to *freq* = $(2^{48}) \times F/f_{CK}$, where F is the desired tuning frequency and f_{CK} is the chip's clock rate as before. The 16-bit phase offset for mixer A is set to *phase* = $(2^{16}) \times Ph/2\pi$, where Ph is the desired phase in radians ranging between 0 and 2π. The phase offset for mixer B is set to *phase* = $(2^{16}) \times Ph/2\pi + (2^{15}) \times F/f_{CK}$. Note that the second mixer phase offset is one frequency step at the sample rate of 2 f_{CK} hence 2^{15} rather than 2^{16} scaling. The configuration software automatically calculates these.

CIC Decimate by cic_dec Filter

The 18-bit mixer output is decimated by a factor of *cic_dec* in the 5-stage CIC[1] filter, where *cic_dec* is any integer between 1 and 256. The value of *cic_dec* can actually be programmed up to 4096 but the gain restrictions normally limit the usable range to 256 (up to 1024 in unusual circumstances). A block diagram of the decimating CIC filter is shown in Figure 9. The input of the CIC decimation filter is equal to the mixer clock rate. The CIC filter has a gain equal to cic_dec^5 that must be compensated for by the *CIC Scale* circuit. This circuit has a gain equal to $2^{cic\_shift-39}$, where *cic_scale* ranges from 0 to 39. CIC gain is $2^{cic\_shift-39} \times cic\_dec^5$ and peak signal gain through *rinf_zpad*, mixer, and CIC must be 1 or less. Normally, this is accomplished by setting the CIC gain itself to 1. The CIC gain can be used to compensate for attenuation earlier in the chain such as receiver interpolation (*gain* = 1/(1+*rinf_zpad*)) or mixer (see the mixer section) or when the input is known not to be full scale. The software sets the gain assuming the input is full scale and compensates for receiver interpolation or mixer gains. The gain analysis is available as an output from the software. When using the software setting *cic_dec* is sufficient – the software computes appropriate settings for the various fields. When programming these manually, one should set *cic_shift* and *ncic*.

[1] Hogenhauer, Eugene V., An Economical Class of Digital Filters for Decimation and Interpolation, IEEE transactions on Acoustics, Speech and Signal Processing, April 1981.

The bit field *cic_sync* controls the precise moment of decimation. The sync can be periodic at any multiple of *cic_dec* without disturbing the processing. If sync is held active, the CIC freezes its output.

The output of the CIC can be attenuated in gain by 6 dB by clearing *cic_rshift*. This appropriate only when *cic_shift* has been set to zero, the signal gain to this point is greater than 0.5, and symmetry is being used in the PFIR filter. In other words, *cic_rshift* should be set to one almost always. The rshift_gain is $2^{cic\_rshift-1}$.

The CIC output data feeding the PFIR needs to be limited to half scale if the PFIR is using symmetry. Control bit field *cic_rcv_full* must be cleared in this case. If the PFIR is not using symmetry, then the data is limited to full scale and the bit field *cic_rcv_full* should be set to one.

The field *cic_rcv* sets the CIC block quiet, receive, or transmit modes.

When the filters are processing both I and Q the data rate out of the CIC filter must not exceed CK/2.

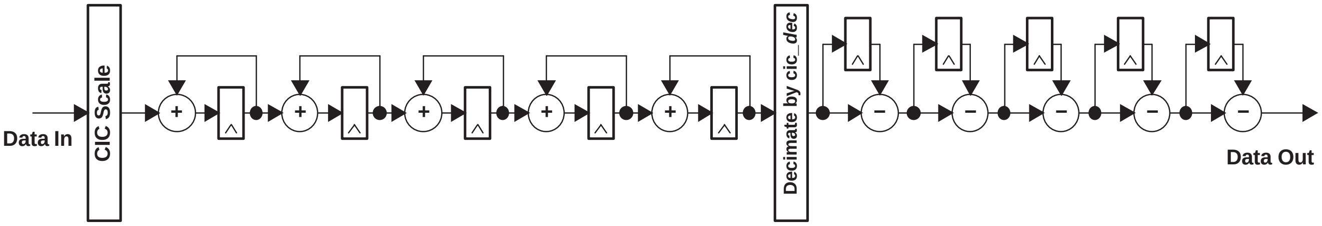


Figure 9. 5-Stage CIC Decimate by *cic_dec* Filter

CIC in Double Rate Mode

Each channel contains two CIC filters (one for I and a second one for Q) allowing input rates of CK complex samples per second. Double rate processing allows input rates of twice this. In this case, the dual CIC's in each channel can be configured to perform as a single CIC at double rate. Thus, channel A CIC can process the I portion of a double rate signal. The time samples (2k+1) come from the I portion of mixer B and are routed to CIC A using the cross receive input (*cic_rcv_cross*). Likewise, channel B CIC processes the Q portion of a double rate signal getting time samples (2k) from the Q portion of mixer A using the cross receive input. This is shown pictorially in Figure 1 with the cross-strapping. When data is input at 2x rate, the CIC must decimate by an even number. The bit fields *cic_2x* and *cic_rcv_cross* must be set when in this mode, either automatically by software or manually. When operating in double rate mode cicA outputs I data only to firA, while cicB outputs Q data to firB. Likewise for C and D when they are operating in double rate mode. This means the PFIR's are operating on real data only (*splitiq* mode).

splitIQ Mode

In some cases, a signal that is input to the chip at CK rate needs to have more filtering capacity than the chip provides in a single channel. As noted above, twice the filtering capacity is available if each filter only has to process I or Q data rather than both. To accomplish this, the user needs to send I data to firA and Q data to firB. Data is mixed in mixA (mixB is idle). This is set automatically in the software by setting *splitiq* to one. It can be set manually by setting *cic_rcv_cross* in cicB, programming mixB to idle, and programming firA and firB to process real signals. Similarly for channels C and D.

Programmable Filter

The decimating programmable finite impulse response (PFIR) filter consists of an input swap RAM to order the incoming data properly, 16 programmable FIR filter cells, a control block and address generator, a final accumulator, and an output gain shift, round, and limit block (see Figure 10 and Figure 19). Each PFIR can process real or complex data.

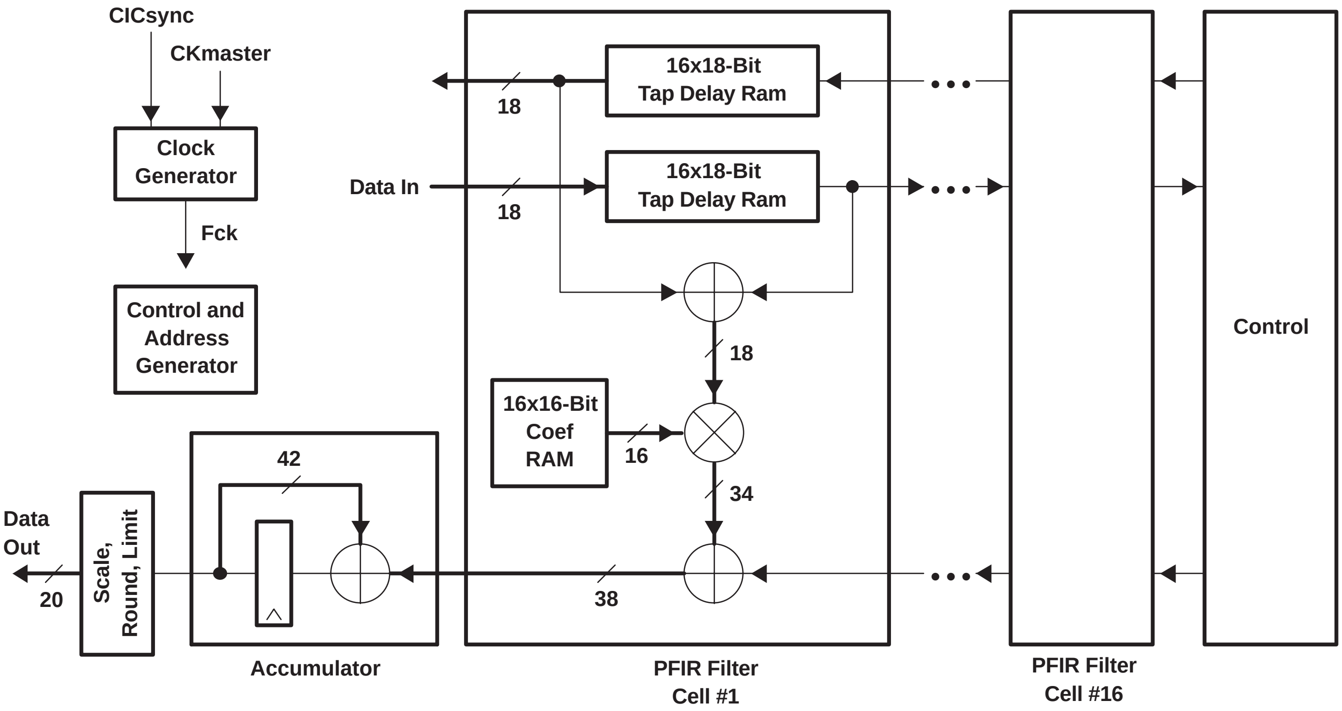


Figure 10. Programmable Filter Block Diagram

Each FIR cell contains a forward 16x18-bit (16 words with 18-bit width) tap delay RAM, a backward 16 x 18-bit tap delay RAM (used for symmetric filters), a pre-adder with 18-bit output (limits the data to 17-bits when using forward and reverse RAMs with symmetric filters), a 16x16-bit filter coefficient RAM, a 16-bit X 18-bit multiplier, and a 38-bit sum chain. The output of the sum chain is sent to an accumulator with 42-bit output and is then shifted up 0−7 bits, round and/or limited with a 20-bit output that is sent to the AGC. The clock generator stops the filter’s clock once an answer is completed and waits until the CIC has provided more data. This simplifies data flow and minimizes power consumption.

The filter row for each channel can be programmed independently.

The filter coefficients can be arranged in banks allowing the user to change between multiple filter sets rapidly and synchronously. Two sets of coefficients might be used in an adaptive application, where one set is being used while the other set is being updated.

For each clock cycle the filter computes 16 taps (32 if symmetric). The number of clocks per output sample is CK/Fsout = *cic_dec* x *fir_dec*. If the data stream is complex then half the clock cycles are used computing the I output and half are used computing the Q output. The tap delay line limits the filter length to 256 if non-symmetric and 512 if symmetric (half this with complex data streams). The maximum number of taps is determined by the cmd5016 program. It can be estimated by:

ntaps = sym x 16 x *fir_dec* x floor (min(*cic_dec*,16/*fir_dec*)/cmplx/*fir_nchan*) − odd

Where:

cmplx = 1 for real data (or *splitiq*) and 2 for complex
sym = 1 for nonsymmetric and 2 for symmetric
odd = 1 for odd, symmetric
fir_nchan = 1 for up and down conversion

If there are multiple filter sets, it is possible that the coefficient storage limits the filter length. The filter supports odd or even symmetry. If the user's filter is significantly shorter than the maximum filter supported, the clock is stopped to the filter block saving power. The filter must be extended using zero coefficients to fill up the full taps used. The software analyzes the taps presented by the user, exploits symmetry if needed for filter length, or if it saves power, zero pads the taps and properly configures the filter. It reports the actual number of taps that the filter uses, since the user could increase their filter length at no additional power. It also reports other possible filter lengths and the corresponding power consumption. A normal design flow would be to start with the desired filter, run the software to see what filter lengths and power levels are possible, then choose the length that best suits the users needs.

Gain for the FIR is sum (coefficients) x $2^{(fir\_shift - 21)}$. The software tries to set the gain so that the overall dc gain to this point is as close to unity as possible, but still less than or equal to 1.

If filter symmetry is exploited, then the preadder is used. Since there is no overflow protection on the preadder, the data entering the filter should be limited to half full scale. Gains should be set so that the gain to the output of the CIC is 0.5 and the output of the CIC filter should be programmed to enforce such a hard limit (*cic_rcv_half*). The software does this if the filter coefficients are symmetric. If the coefficients are symmetric and the user does not wish the hardware to exploit this, adding a zero to the end of the coefficients prevents the software from recognizing the coefficient symmetry.

The filter block can be configured to process real or complex data. The decimation can be programmed from 1 to 16. Only real coefficients are supported by the software (software support for complex coefficients is a possible future enhancement). The filter configuration is determined from software inputs *fir_dec*, *cic_dec*, and the filter coefficient file. Manually programming the filter is not recommended. *fir_shift* is normally calculated to provide unity gain to this point in the processing chain but can reasonably be overridden by the user.

Explaining the full operation of the filter such that it could be manually programmed is beyond the scope of this document. The software is the best way to configure the filter.

Power Meter

The data is then sent to a power meter. The power meter squares the top 12 bits of the data, keeps the top 17 bits of the result, and integrates it for up to 2^{16} words, transferring the 32-bit result to a pair of control registers. Handshaking is provided to let the user know when data is ready. Note that the integration is over a number of words so if the data is complex the number of samples integrated is one half the number of words. If the filters are configured in a *splitiq* mode then the power meters of the real and imaginary channels needs to be combined in the user's software. A sync is available to start the power measurement period. The power meter automatically starts a new measurement at the conclusion of the last one. The contents of the power meter registers should be considered unstable from eight clocks after syncin to eight clocks plus on output sample time. (The actual unstable time is around 0.5 ns, so even reading during this window provides correct answers most of the time.) Reading during data transfer results in an erroneous output (some bits being updated, while others are not) but does no other harm. A periodic power measurement with simple reads can be safely made if software is sufficiently aware of time to avoid this interval. Otherwise, software can use the handshake signal to ensure reliable power measurements. If the processor is not sufficiently aware of time and the user wishes to avoid using the handshake, one could also read the power meter twice in rapid succession checking that the same value is read both times to be certain a valid power level is read. Figure 11 shows the hardware.

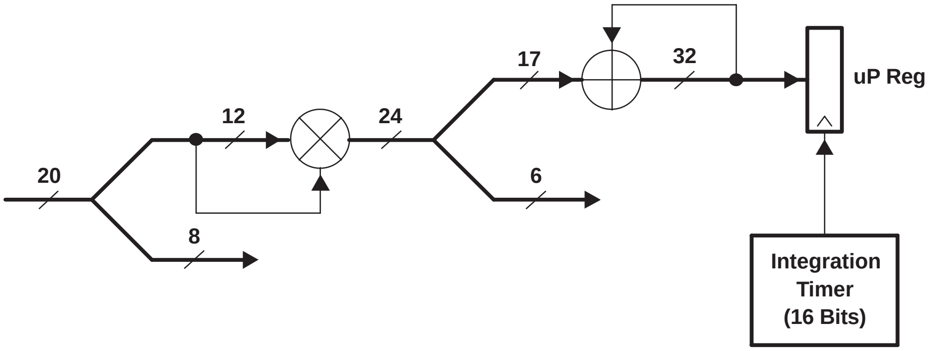


Figure 11. Power Meter Hardware

When the integration counter is synced (*pwr_mtr_sync*) and again whenever it reaches terminal count (*pwr_mtr_integ*), a done signal is given. The done signal that comes from syncing the integration counter should be discarded (using the periodic sync counter to sync the integration counter is not recommended). On done, the accumulator value is strobed into the registers (page 0x13 address 0x1a and 0x1b), the ready bit (page 0x13 address 0x1c bit 15) is set, and the accumulator is cleared. Note that there are four independent power meters, the addresses here are for channel A (channels B, C, and D are at the same address but on page offsets of 0x20, 0x40, and 0x60 respectively).

The control bus and system clock are at different rates, with either one being faster or slower than the other. In most cases, the system clock is the fastest. To get the control bus to the system clock domain, a one shot is used. Firing the one shot clears the ready bit and lets the chip know the power was read. There are two ways to fire the one shot. Automatically, when the msb of the power is read page 0x13 address 0x1c bit 10 = 1 or manually by writing a 0 (arming) and then a 1 (firing) to page 0x13 address 0x1c bit 11 (P0x13A0x12B10 must be 0). There needs to be two system clocks between writing the 0 and writing the 1, also two clocks after writing the 1, before rearming.

There are two status bits, *too_soon* B13 and *missed* B14. If the one shot is fired when the ready bit B15 is low, then *too_soon* is set. The user must reset it. If done happens when the ready bit is set, the missed bit is set. Again, it is reset by the user.

Example using msb to fire one shot:

1. Sync integration counter
2. Wait for ready bit to be 1 (8 clocks or less depending on sync source)
3. Read msb of power (also fires one shot to clear ready bit) and ignore it.
4. Wait for ready bit to be 1
5. Read power lsb
6. Read power msb
7. Check to be sure missed bit is not set
8. Go to step 4

NOTE: The *too_soon* bit is never set if ready is active when msb is read.

Example using manual one shot firing:

1. Sync integration counter
2. Wait for ready bit to be 1 (eight clocks or less depending on sync source)
3. Arm and fire one shot to clear ready
4. Wait for ready bit to be 1
5. Read power lsb
6. Read power msb
7. Arm and fire one shot to clear ready
8. Check to be sure missed bit is not set
9. Go to step 4

NOTE: The *too_soon* bit is never set if ready is active when one shot fires.

Automatic Gain Control (AGC)

The GC5016 automatic gain control circuit is shown in Figure 12. The basic operation of the circuit is to multiply the 20-bit input data from the PFIR by a 19-bit gain word that represents a gain or attenuation in the range of 0 to 128. The gain format is mixed integer and fraction. The 7-bit integer allows the gain to be boosted by up to factor of 128 (42 dB). The 12-bit fractional part allows the gain to be adjusted up or down in steps of one part in 4096 or approximately 0.002 dB. If the integer portion is zero, then the circuit attenuates the signal.

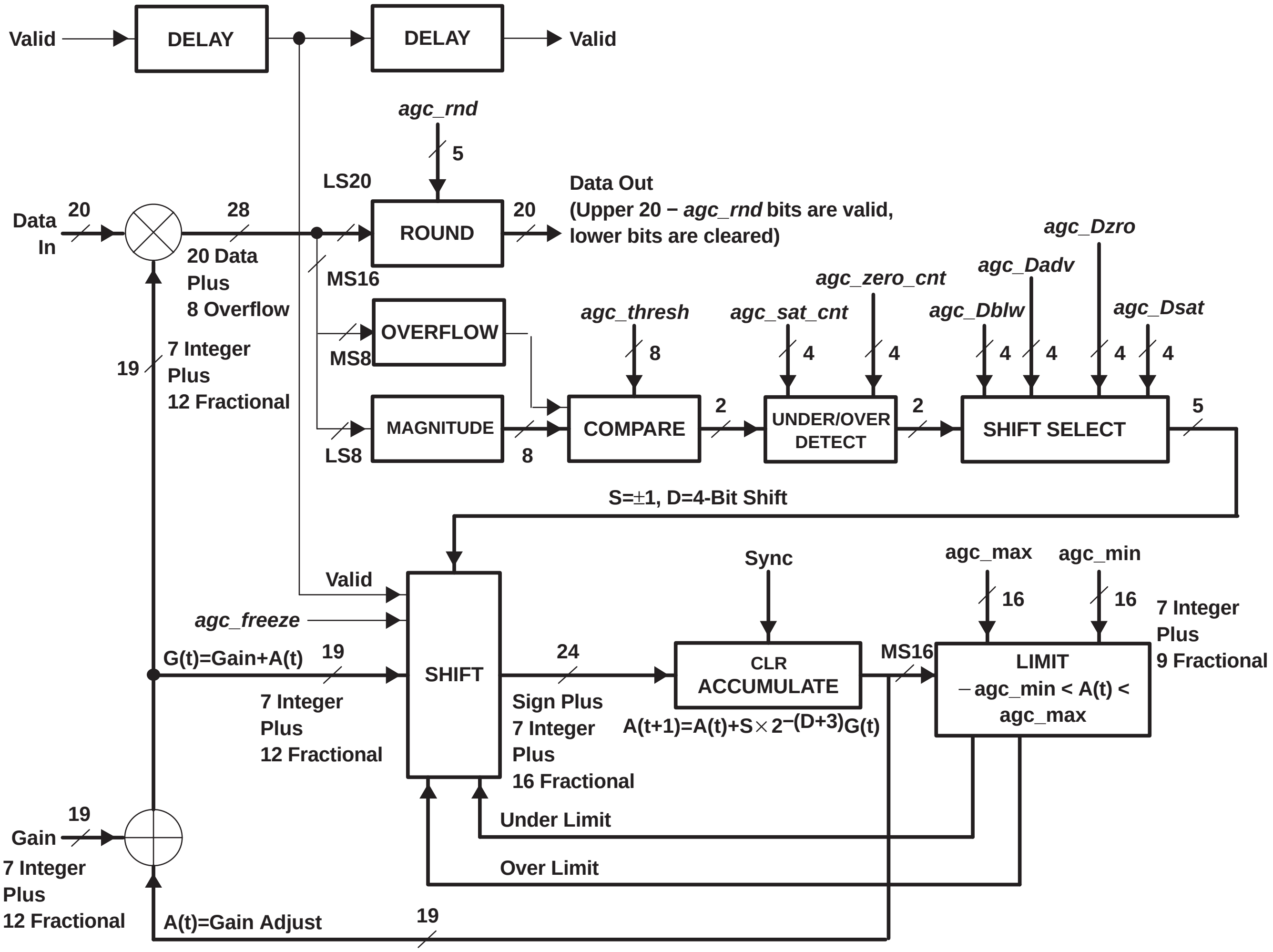


Figure 12. GC5016 AGC Circuit

The gain adjusted output data is saturated to full scale and then rounded to between 4 and 20 bits in steps of one bit.

The AGC portion of the circuit is used to automatically adjust the gain so that the median magnitude of the output data matches a target value, which is performed by comparing the magnitude of the output data with a target threshold. If the magnitude is greater than the threshold, then the gain is decreased, otherwise it is increased. The gain is adjusted as: G(t) = G + A(t), where G is the default, user supplied gain value, and A(t) is the time varying adjustment. A(t) is updated as A(t) = A(t) + G(t) x S x $2^{-(D+3)}$, where S=1 if the magnitude is less than the threshold and is −1 if the magnitude exceeds the threshold, and where D sets the adjustment step size. Note that the adjustment is a fraction of the current gain. This is designed to set the AGC noise level to a known and acceptable level, while keeping the AGC convergence and tracking rate constant, independent of the gain level. Because the adjustment is a fraction of the current gain, one can show that the AGC noise is an amplitude jitter in the data output equal to ±(data output) x $2^{-(D+3)}$. This means that the AGC noise is always 6 x (D+3) dB below the output signal's power level. The AGC attack and decay rate is exponential with a time constant equal to $2^{(D+1.75)}$ complex samples. This means the AGC covers to within 63% of the required gain change in one time constant and to within 98% of the change in the four time constants.

If one assumes the data is random with a Gaussian distribution, which is valid for UMTS if more than 12 users with different codes have been overlaid, then the relationship between the RMS level and the median is MEDIAN = 0.6745 x RMS, hence the threshold should be set to 0.6745 times the desired RMS level.

The step size can be set using four values of D. The user can specify separate values of D for when the magnitude is below threshold (*agc_Dblw*), is above threshold (*agc_Dabv*), is consistently equal to zero (*agc_Dzro*), or is consistently equal to maximum (*agc_Dsat*). This allows the user to set different attack and decay time constants and to set shorter time constants for when the signal falls too low (nearly zero) or is too high (saturates). The magnitude is considered to be consistently nearly zero by using a 4-bit counter that counts up every time the 8-bit magnitude value is nearly zero and counts down otherwise. Nearly zero is defined by and'ing the magnitude with a zero mask before checking to see if it is zero. If the counter's value exceeds a user specified threshold, then *agc_Dzro* is used. Similarly, the magnitude is considered too high by using a counter that counts up when the magnitude is maximum and counts down otherwise. If this counter exceeds another user specified threshold, then the *agc_Dsat* is used.

The AGC is also subject to user specified upper and lower adjustment limits. The AGC stops incrementing the gain if the adjustment exceeds *agc_max*. It stops decrementing the gain if the adjustment is less than −*agc_min*. The *agc_max* and *agc_min* bits are 16-bit values which line up with the most significant 16 bits of *gain_msb* and *gain_lsb*.

The input data is received with a valid flag that is high when a valid sample is received. For complex data, the I and Q samples are processed as if they were two real samples. An adjustment is made for the magnitude of the I sample, and then another adjustment is made for the Q sample.

Fixed Gain Control

The AGC can be turned off by setting the *agc_freeze* control bit. The AGC adjustment loop is cleared using the *gain_sync* control bit field. A static gain is set by setting G0 using the *gain_lsb* and *gain_msb* bit fields, by setting *agc_freeze*, and by setting *gain_sync* to be always active. The *gain_sync* control can also be used to synchronize gain changes across multiple channels or across multiple chips.

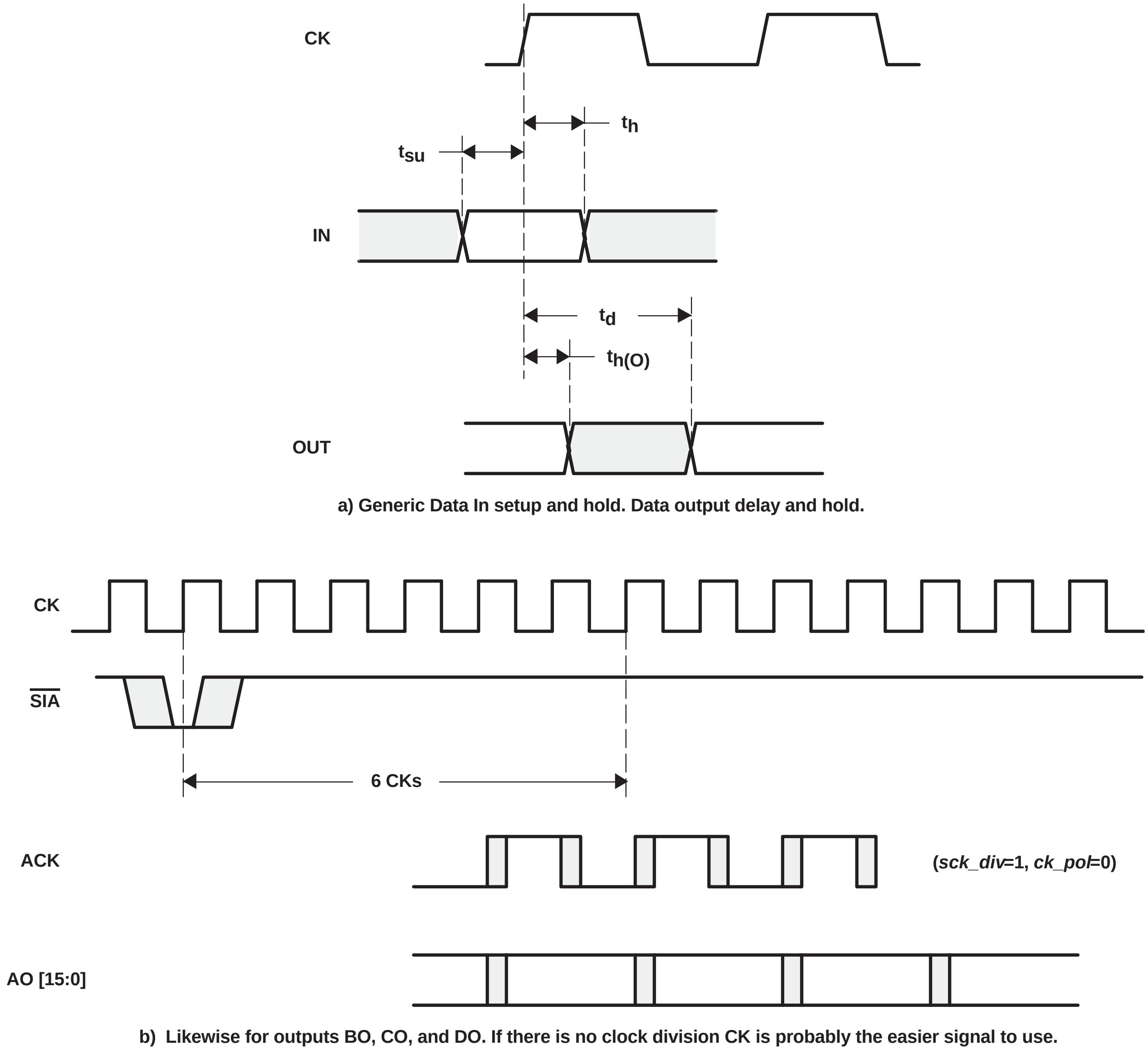
Receive Output Interface

This section details the output interface when the GC5016 is in down-conversion mode, which is typically with an FPGA or ASIC processor.

The GC5016 has four 16-bit output ports. The output interface consists of 16 parallel output pins, a clock, and a frame strobe. The parallel output data pins for the GC5016 are AO[15..0], BO[15..0], CO[15..0], and DO[15..0]. The letters A..D refer to the four separate channels A..D. Separate port clocks [A..D]CK and frame strobes [A..D]FS are provided by the GC5016 for each output port.

The clock [A..D]CK for each port are generated by dividing the GC5016's main clock CK by a programmable divider for each port. This allows for output rates for each port lower than the GC5016's main clock rate. The clock dividers can be synchronized by the methods described in the Synchronization section*.* The polarity of each divided port clock [A..D]CK is user programmable. The input sync signal needs to be set up for t_{su} before CK rising and held for t_{hd} after CK rising. The outputs retains their previous value for $t_{h(O)}$ and has the new data valid t_d after CK rising as shown in Figure 13(a).

The divided port clocks [A..D]CK are treated internal to the GC5016 as data signals and hence change nearly simultaneously with the output data pins AO[15..0], BO[15..0], CO[15..0], and DO[15..0] (typically 0.5 ns later due to the xor gate for clock polarity). When *ck_pol* is 0 data transitions just before the rising edge of [A..D]CK – in this case the falling edge of [A..D]CK should be used. (If *ck_pol* is 1, then the rising edge should be used). Since the timing of the divided clocks is known from the sync pulse, it is also reasonable to sample the data using CK and use the sync information to identify which sample should be kept. Figure 13(b) shows example timing with a clock divide of two (*sck_div*=1) and *ck_pol*=0. Sync initiates the clock divider and frame counter. The serial clock output is defined starting six clocks after the incoming sync. At that point, the serial clock is high for the user defined time and low for the same time (assuming the clock is 50% duty cycle). The serial clock then repeats. The rising edge of sck is always one output delay after a clock rising edge. If the clock division is even, then the falling edge of sck is after a clock rising edge. If the clock division is odd, the falling edge of sck is after a clock falling edge (to make the clock output duty cycle approximately 50%). The sync may be one time or repetitive in any multiple of (1+*sck_div*). There must be enough sck's to allow the data to exit the chip. The configuration software checks this constraint. The frame strobe always aligns with sck. The frame strobe is periodic if sck rate modulo output sample rate is zero; otherwise the frame period varies between X sck periods and X+1 sck periods. Data can be sampled on any rising edge of CK and is stable for the entire time, except for a short time after the critical CK rising edge. Starting an output hold time after the critical CK rising edge to an output delay time after the same edge AO, AFS, and ACK outputs are changing. ACK is approximately 50% duty cycle if CK is 50% duty cycle. The user may safely use the falling edge of ACK to sample the data or the user may use CK (and then decimate down). Likewise for outputs BO, CO, and DO. If there is no clock division CK is probably the easier signal to use.



a) Generic Data In setup and hold. Data output delay and hold.

b) Likewise for outputs BO, CO, and DO. If there is no clock division CK is probably the easier signal to use.

Figure 13. Example Timing Diagram for the Down-Conversion Channel Output Ports

The frame strobes [A..D]FS are used to signify the beginning of a data frame for each port. The frame strobes are set high by the GC5016 with the first word in a frame.

The GC5016 may be configured to have one port for each channel. The number of pins used for a port is user programmable as 4, 8, or 16 pins. The number of bits in a word is user programmable as 4, 8, 12, 16, or 20 bits. When the number of word bits is larger than the number of pins, the data is sent time domain multiplexed at the divided port clock rate [A..D]CK. The most significant bits (MSBs) are sent first. For complex data, I is followed by Q. The frame strobe is set high with the MSB of the first I word as shown in Figure 14. For example, with no TDM, IQ multiplexed, 16 bits over four pins there are eight transfers, so (*cic_int* x *fir_int*/(*sck_div*+1) must be greater or equal to eight.

The GC5016 can round the output to any size from 4-bits to 20-bits and supplies zeros for the extra LSB's. The number of bits after rounding can be smaller than the number of bits in a word.

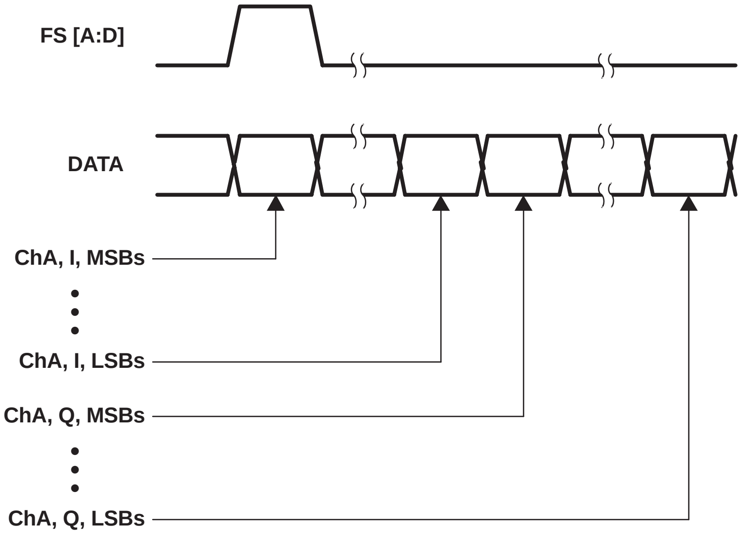


Figure 14. Time Multiplexed Data Sequence for a Single Channel

The user can output either real or complex data.

splitIQ Mode

In the case where a complex signal is split, so that the I and Q are processed in different channels, the outputs should be configured to output real data and the I portion is output from one port and the Q from another. Any of the previously mentioned formats can be used.

Embedded AGC

The output formatter can be set up to output AGC information together with the data. This is appropriate for systems that require both AGC and rapid, precise knowledge of the actual gain. This is done by configuring the round to 8 bits (or less), while configuring the port to support 16 bits. The lower 8 bits of the I and Q word are then replaced by the AGC information as shown in Table 2. The most significant 14 bits of gain are output together with 2 bits of state information. The state information shows whether the AGC is in a zero or max state that use the faster gain adaption constants. The embedded AGC only works for total decimation rates (*cic_dec* x *fir_dec*) of 5, 6, and 10 or more. The cmd5016 program issues a warning if this mode is enabled for total decimations of 1, 2, 3, 4, 7, 8, or 9. The embedded AGC only works for total decimation rates (*cic_dec* x *fir_dec*) of 5, 6, and 10 or more. The cmd5016 program issues a warning if this mode is enabled for total decimations of 1, 2, 3, 4, 7, 8, or 9.

Table 2. Bit Placement for Gain Output With Data

PINS	TIME	CONTENT
AO 15..8	0	I 7..0
AO 7..0	0	Gain 18..11
AO 15..8	1	Q 7..0
AO 7..2	1	Gain 10..5
AO 1	1	Zero signal state
AO 0	1	Saturated signal state

Multichannel Time Division Multiplex

Two or four channels can share one port, with the data for each channel time multiplexed. In this case, the frame strobe is set high with the MSB in the first I word in the frame as shown in Figure 15. The number of pins used for a port is user programmable as 4, 8, or 16 pins and the number of bits in a word is user programmable as 4, 8, 12, 16, or 20 bits. The embedded AGC information format can also be combined with this mode.

NOTE: The receive TDM output is sent out the D port in the sequence D, C, B, and A. Four channels are always output even if not all channels are active.

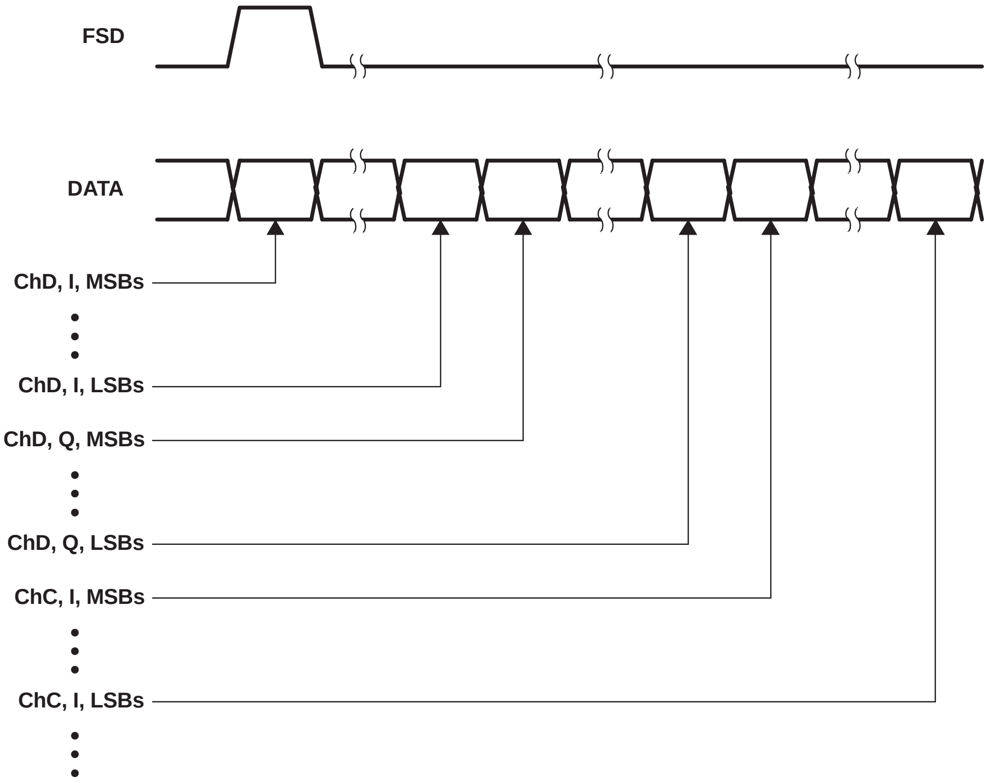


Figure 15. Time Multiplexed Data Sequence for Multiple Channels Per Port

Global register (0x3) provides four control bits to independently enable each of the data output ports AO[15..0], BO[15..0], CO[15..0], and DO[15..0]. Four more control bits enable the port clocks [A..D]CK, and frame strobes [A..D]FS for each of the four ports as explained in Table 17.

Overall Gain in Receive Mode

The overall gain in the receive mode is a function of zero padding (*rinf_zpad*), the CIC decimation (*cic_dec*), the cic shift settings (*cic_shift* and *cic_rshift*), the sum of the programmable filter taps (PFIR_SUM), the filter output shift (*fir_shift*) and the final gain in the agc circuit (G). The cmd5016 program, described later in the data sheet, sets the *cic_shift*, *cic_rshift*, *fir_shift,* and G to their optimum levels for a targeted overall gain.

The overall gain is:

DDC_gain = zpad_gain x mix_gain x cic_gain x rshift_gain x fir_gain x agc_gain

Where the individual gains are:

zpad_gain = 1 / (*rinf_zpad*+1)

mix_gain = 1/2

cic_gain = cic_dec^{5} x $2^{(cic\_shift-39)}$

rshift_gain = $2^{(\mathrm{cic\_rshift}-1)}$

fir_gain = PFIR_SUM x $2^{(fir\_shift-21)}$

agc_gain = G / 4096

The restrictions on the gain settings are:

1. To prevent overflow in the CIC, *cic_shift* must be set such that:

   zpad_gain x mix_gain x cic_gain ≤ 1

2. If *rinf_zpad* is greater than *cic_dec*, then *cic_shift* must be set such that:

   (1/*cic_dec*) x mix_gain x cic_gain ≤ 1

3. For symmetric filters the maximum amplitude allowed into the fir is one-half, so *cic_shift* must be set such that:

   zpad_gain x mix_gain x cic_gain x rshift_gain ≤ 1/2

4. The *cic_rshift* control is set to 1 (this control is only used to extend the allowable *cic_dec* range, and must be used with care).

The fir_gain and agc_gain are used to adjust the overall gain to match the user’s desired gain (overall_gain). The *fir_shift* control should be set such that:

zpad_gain x mix_gain x cic_gain x rshift_gain x fir_gain ≤ overall_gain

and the final agc_gain is set to give the desired overall_gain.

This equation gives unity gain for dc or complex data inputs. For real inputs, such as from an ADC, the DDC_gain is typically set to 2 (6 dB). The gain of 2 compensates for the loss of 6 dB when tuning a signal to dc and filtering out the negative image. Mathematically this is illustrated by using a an example input signal s(t) modulated up to a frequency of ”ω”. The input is defined as:

$d(t) = s(t) \times \cos(\omega t) = s(t) \times (e^{j\omega t} + e^{-j\omega t}) / 2$

If this is mixed down to dc using $e^{-j\omega t}$, the result is:

$y(t) = d(t) \times e^{-j\omega t} = s(t) \times (1 + e^{-j2\omega t}) / 2$

Which, after low-pass filtering becomes:

$y(t) = s(t) / 2$

Hence, a gain of 2 is required to bring s(t) back to full scale.

GC5016 UP-CONVERSION MODE

The up-conversion mode of the GC5016 is implemented using four channels that perform input formatting, gain, and programmable filtering. The data is then sent to a dual CIC block that either interpolates an I and Q signal up to CK rate or interpolates just a real (or just an imaginary) signal up to 2 x CK rate. Data is then sent to a complex mixer to place the signal at the desired carrier. The CIC and mix blocks are organized in pairs (two dual CIC's and two complex mixers) to facilitate double rate mode.

The up-conversion channel details are reproduced in Figure 16. The up-conversion channel accepts 20-bit real or complex input data, multiplies by a gain, interpolates by 1 to 16 in a programmable FIR filter (PFIR), interpolates by 1 to 256 in a 6-stage cascaded integrator-comb filter (CIC), multiplies by a complex, programmable numerically controlled oscillator (NCO), and outputs 21-bit data. Each part is described in detail below for the most common configuration, where each up-conversion channel processes one complex signal. Other configurations, such as processing where I and Q are processed by separate up-conversion channels are described later.

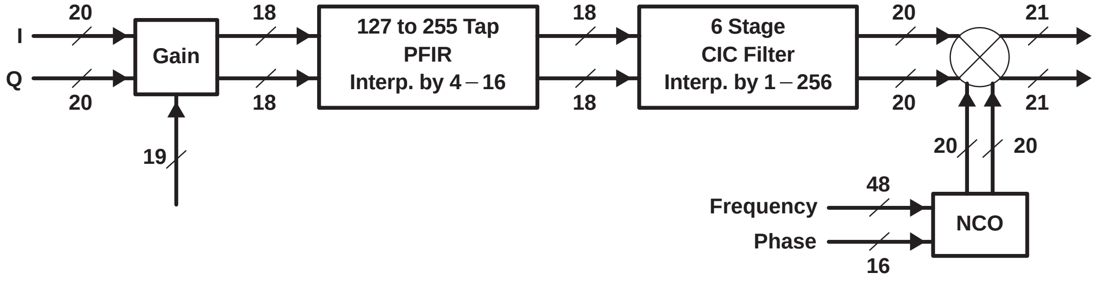


Figure 16. Up-Conversion Channel Detail

Transmit Input Interface

This section details the input interface when the GC5016 is in up-conversion mode, which is typically with an FPGA or ASIC processor.

The GC5016 has four input ports A−D. Each input port consists of an output frame strobe, 16 parallel data inputs, and an output clock. The frame strobes [A..D]FS are used to signify the beginning of a data frame. The parallel input data pins for the GC5016 are AI[15..0], BI[15..0], CI[15..0], and DI[15..0], where the letters A..D refer to the four separate up-conversion channels A..D. For some applications the output clocks generated by the GC5016 are useful – most applications do not use them.

Frame Strobe

Each channel has its own frame strobe generator. The beginning of the incoming frame is internally identified by a user programmable (*tinf_fs_dly* in divided clocks) delayed version of this strobe. The delay allows for turn-around time in handshaking with external devices such as an FPGA or ASIC. Note that whatever is present on the pins at that time is taken as the incoming data. Several different formats are acceptable for the incoming data as discussed in the next sections. The delay is implemented using a counter that is reset with each frame sync going out. This limits the available delay to the frame length. Since the frame strobe is strictly periodic, longer turn around times are feasible by programming the delay to *tinf_fs_dly* = (desired turn around time) modulo *frame_length*.

Real or IQ Multiplexed

The GC5016 may be configured to have one input port for each channel. The number of pins used for a port is user programmable as 4, 8, or 16 pins. The number of bits in a word is user programmable as 4, 8, 12, 16, or 20 bits. When the number of word bits is larger than the number of pins, the data is sent time domain multiplexed at the divided port clock rate [A..D]CK. The most significant bits (MSBs) are sent first. For complex data, I is followed by Q. The frame strobe is set high with the MSB of the first I word (see Figure 17) when *tinf_fs_dly* = 1. For other word sizes the user is required to supply zeros to the unused pins for the extra LSB's. For example, a 5-bit word would be input on the five most significant pins, the lower three would need to be 0, and the lower eight if unused should not be left floating.

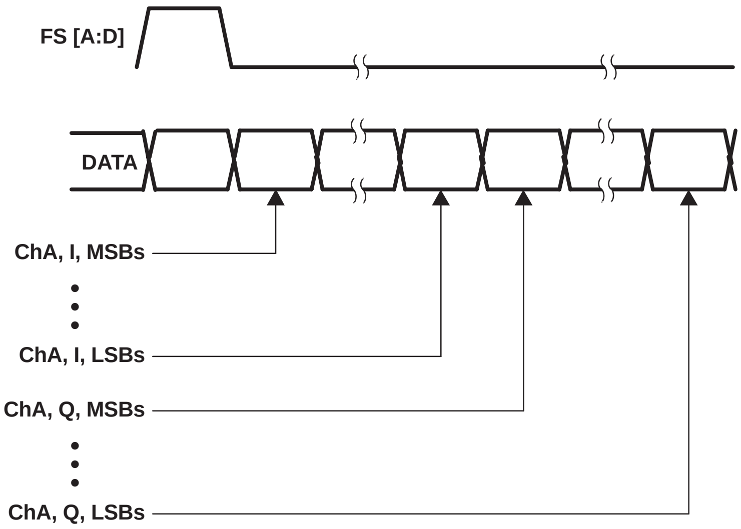


Figure 17. IQ Multiplexed Data Sequence for a Single Channel

IQ Parallel

The GC5016 may also be configured to accept input data with 8 bits of I and 8 bits of Q in parallel. I msb should be placed at AI[15] and Q msb at AI[7] (or BI, CI, or DI). This mode is only valid for 8-bit I and 8-bit Q using 16 pins. For applications with fewer bits unused, lsb's should be grounded. There is no corresponding output mode for down conversion.

Multichannel Time Division Multiplex

The A port may also be used as the source for all channels configured in transmit. In TDM mode, data for channel A is followed by data for channel B. In four channel TDM mode, the channel order is A, B, C, D. The frame strobe is set high with the MSB in the first I word in the frame (see Figure 18), when *tinf_fs_dly* = 1. As above, the number of pins used for a port is user programmable as 4, 8, or 16 pins and the number of bits in a word is user programmable as 4, 8, 12, 16, or 20 bits. TDM may be combined with the IQ parallel mode to allow one channel per clock in TDM mode.

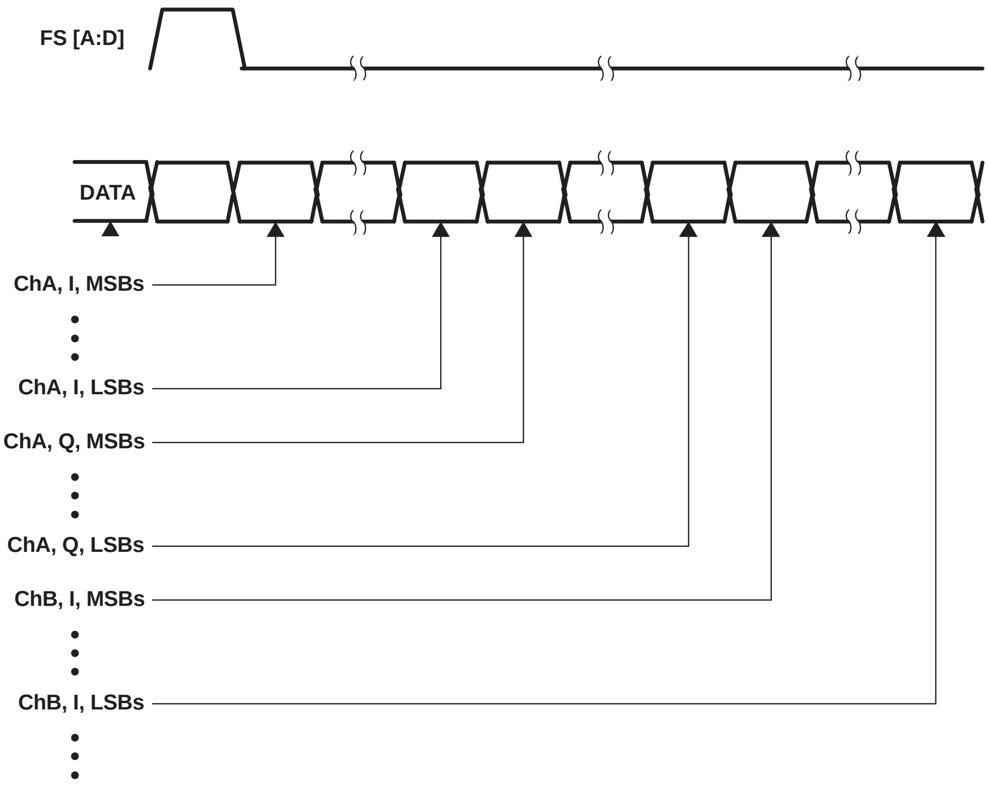


Figure 18. Time Multiplexed Data Sequence for Multiple Channels Per Port

splitIQ Mode

For applications using complex inputs and either increased PFIR filtering or double rate outputs the *splitiq* mode should be used. In this mode, the real portion of the input data should be entered on channel A and the imaginary portion on channel B (or channels C and D). In this mode, a maximum of two transmit channels can be supported by one chip. The configuration software automatically enables *splitiq* if the output rate is double.

The transmit input formatter accepts the various input formats, zero pads the lsb’s to create a 20-bit word, and passes the data onto gain.

Clocking

The incoming data is clocked on the chip master clock CK. The chip can be programmed to grab data every second, third, etc., clock edge allowing the data source to supply data at a lower speed and easing signal integrity issues. The user controls the clock division using *sck_div*. A value of 0 means that every clock edge is used, a value of 1 means that every other clock edge is used, etc. The clock division phasing is controlled by a general sync (*sck_sync*). The sixth clock edge after sync is used and every (*sck_div*+1) edge thereafter. The time (in CK clocks) between data frames is the product of PFIR interpolation (*fir_int*) and CIC interpolation (*cic_int*). The divided clock must divide this evenly so (*cic_int* x *fir_int*) % (*sck_div*+1) must be 0 for the framing to be fixed length (otherwise the length varies between two values). There must be enough time to get the data into the chip. There are (*cic_int* x *fir_int*)/(*sck_div*+1) divided clocks available for the data transfer. This must be long enough for the entire data frame being (bits/pins) x (2 if complex) x (nchannels if TDM). The software checks these constraints.

The clock outputs [A..D]CK are used primarily in receive but may be of use in some transmit applications – either as a data bit to indicate when data should be valid or in low frequency applications as a clock. They are generated by dividing the GC5016’s main clock CK by programmable dividers *sck_div*+1 for each channel. The input data transfer clock rate is then CK/(*sck_div*+1). The clock dividers can be synchronized by the methods described in the Synchronization section. The polarity of each divided port clock [A..D]CK is user programmable. For many applications, the input data transfer clock rate is the same as the main clock CK. In this case, the output [A−D]CK should be ignored. For all other applications [A..D]CK is redundant with the information provided in synching. The user may either use synching and ignore [A..D]CK or can treat [A..D]CK as a data bit to indicate when data must be valid. In a few applications, [A..D]CK is used as a clock (when interfacing to slow FPGA's for example). Note that the [A..D]I setup time relative to [A..D]CK is relatively large (hence the recommendation to restrict their use as a clock to low frequencies).

The outputs [A..D]CK are delayed relative to CK by a normal clock to data valid. The input data is latched on the rising edge of the main clock CK and the input data is registered on the first CK that the divided port clock [A..D]CK is high. Therefore, the input data pins AI[15..0], BI[15..0], CI[15..0], and DI[15..0] must be valid for $t_d + t_{su}$ ns prior to the rising edge of the divided port clock) before the change in the divided clocks [A..D]CK (see Figure 19).

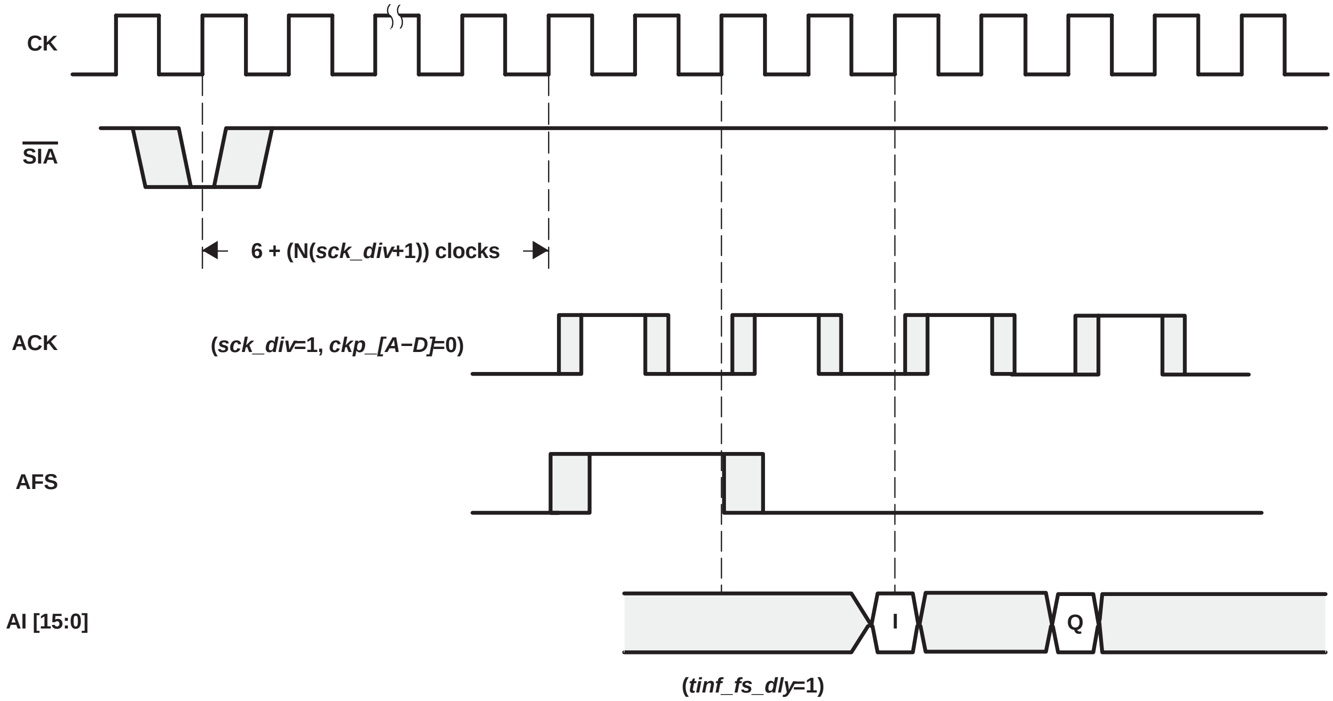


Figure 19. Example Timing Diagram for Transmit Input Ports

Gain

Each 20-bit input sample is multiplied by a 19-bit gain word. The 16 LSB's are stored in one register *gain_lsb* and three MSB's in another *gain_msb*. The gain adjustment is GAIN/2^{12}, where the gain word (*gain*) ranges from 0 to (+2^{19} – 1). Negative gains are not allowed. This gives a 0.002-dB gain adjustment resolution. Setting *gain_msb* and *gain_lsb* to zero clears the channel input. A different gain can be specified for each channel. The gain values are double buffered and are transferred to the active register at the first I sample after sync (*gain_sync*). The gain block for each up-conversion channel contains a dedicated 20x20 multiplier to apply fine gain control. The result rounded to 18 bits and limited to one for non-symmetric filters or one-half for symmetric filters and sent to a programmable filter. This is controlled manually using *gain_half* or the software automatically calculates it.

Programmable Filter

The interpolating programmable finite impulse response (PFIR) filter consists of an input swap RAM to order the incoming data properly, 16 programmable FIR filter cells, a control block and address generator, a final accumulator, and an output gain shift, round, and limit block (see Figure 20). Each PFIR can process real or complex data.

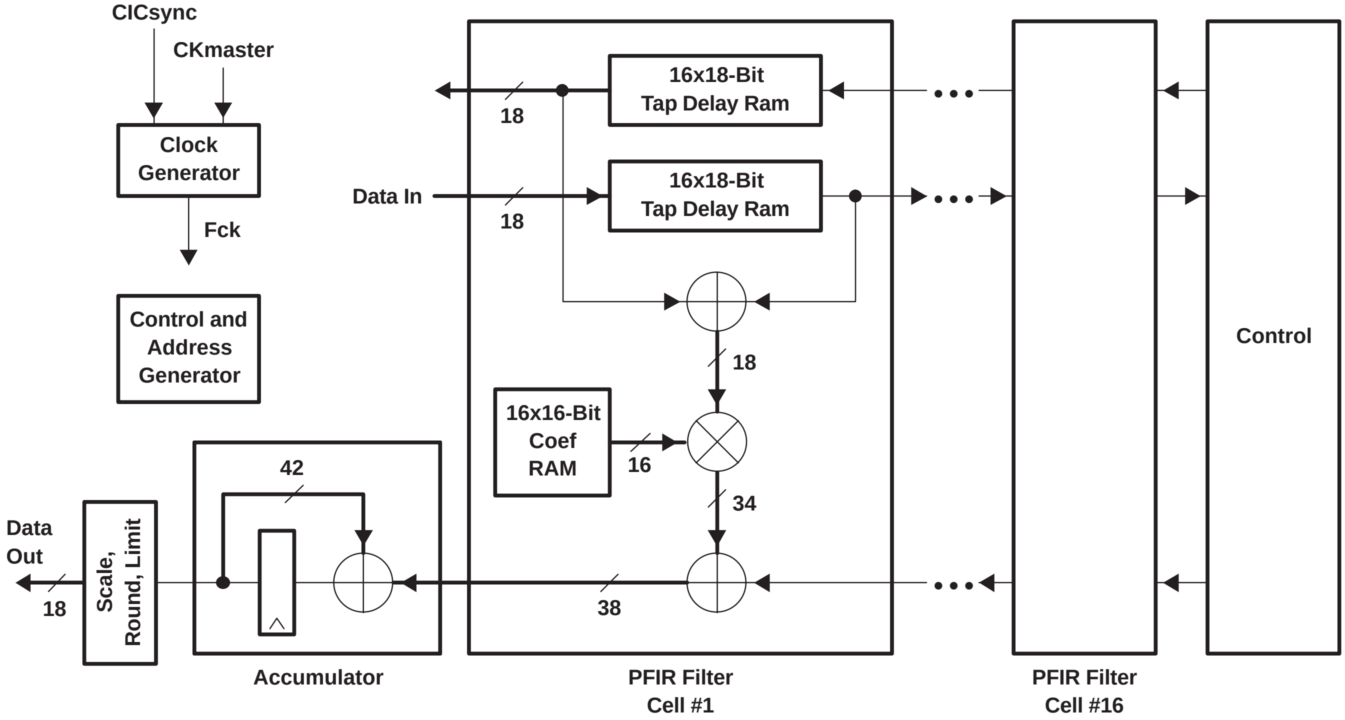


Figure 20. Programmable Filter Block Diagram

Each FIR cell contains a forward 16x18-bit (16 words with 18-bit width) tap delay RAM, a backward 16x18-bit tap delay RAM (used for symmetric filters), a pre-adder with 18-bit output (limits the data to 17-bits when using forward and reverse RAMs with symmetric filters), a 16x16-bit filter coefficient RAM, a 16x18-bit multiplier, and a 38-bit sum chain. The output of the sum chain is sent to an accumulator with 42-bit output and is then scaled, round, and/or limited with an 18-bit output that is sent to the CIC filter. The clock generator stops the filter’s clock once an answer is completed and waits until the CIC has consumed the data. This simplifies data flow and minimizes power consumption.

The filter row for each channel can be programmed independently.

Configuring the PFIR is a difficult task if done manually. A software configuration program should be used instead. The program reads in the desired configuration, filter taps, and properly configures the filter. The chip reduces power for shorter filters. The program informs the user of possible filter lengths and corresponding power consumption. The program extends the filter with zeros to round up the filter length to the next possible size. If the coefficients are symmetric the software programs the hardware to exploit it. If the coefficients are symmetric and the user does not want the hardwire to exploit this then a zero should be added to the coefficients either at the front or back.

For each clock cycle, the filter computes 16 taps (32 if symmetric). The number of clocks is *cic_int*. If the data stream is complex, then half the clock cycles are used computing the I output and half are used computing the Q output. The coefficient storage limits the filter length to 256 if non-symmetric and 512 if symmetric. The maximum number of taps is determined by by the cmd5016 program. It can be estimated by:

ntaps = sym x min(256,16 x *fir_int* x floor(*cic_int*/cmplx/*fir_nchan*)) − odd

Where:

fir_nchan = number of channels in one FIR, always one for up or down conversion
cmplx = 1 for real data (or *splitiq*) and 2 for complex
sym = 1 for nonsymmetric and 2 for symmetric
odd = 1 for odd, symmetric

NOTE: For interpolation, even and odd symmetry are available if interpolating by 1, only odd symmetry is available if interpolating by 2, and higher interpolations cannot exploit symmetry.

The chip can be configured to provide additional filtering resource by either operating at a higher clock rate (and using transmit output hold to keep the output sample rate constant) or by using *splitiq*. The *splitiq* mode uses one filter to process the real portion of a complex data stream and a second filter to process the imaginary portion. Using *splitiq* cuts the number of channels available in half and doubles the number of taps. The filter gain is PFIR_SUM / (*fir_int* x $2^{21-\mathit{fir\_shift}}$).

Dual CIC Filter

The 18-bit output from the PFIR is interpolated by a factor of *cic_int* in the 6-stage CIC filter, where *cic_int* is any integer between 1 and 4096. The value of *cic_int* is programmed independently for each channel. A block diagram of the CIC filter is shown in Figure 20. The output of the CIC interpolation filter is equal to the mixer clock rate. The CIC filter has a gain equal to $\mathit{cic\_int}^5$ that must be removed by the scale and round circuit. This circuit has a gain equal to $2^{-41+\mathit{cic\_shift}}$, where *cic_shift* ranges from 0 to 39. Overall CIC gain is $2^{-41+\mathit{cic\_shift}}$ x $\mathit{cic\_int}^5$ and should normally be set to be 1 or less. For CIC interpolations above 294 the gain exceeds one even with the *cic_shift* set to zero. The CIC can be programmed to be five stages rather than six stages, reducing the overall gain to $2^{-41+\mathit{cic\_shift}}$ x $\mathit{cic\_int}^4$ and allowing *cic_int* to go up to 1217 before gain exceeds unity. Values of *cic_int* exceeding 1217 are not generally usable due to the excessive CIC gain. The configuration software uses *cic_int* to calculate the appropriate control settings for *cic_shift*, *cic_xmt_5stg*, *cic_xmt_d6stg*, *cic_2x*, and *ncic* (*ncic* is *cic_int*−1 for normal cases and *cic_int*/2−1 for double rate).

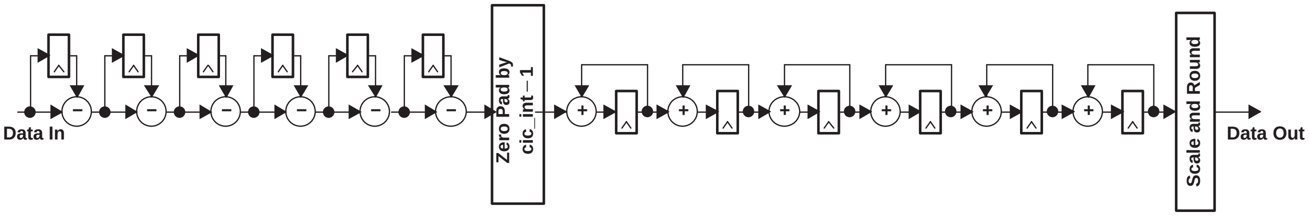


Figure 21. 6-Stage CIC Interpolate by *cic_int* Filter

The CIC filter must be initialized when the GC5016 is first configured or whenever the interpolation value *cic_int* or the shift value *cic_shift* are changed. The CIC filter is initialized using the flush controls described below. If the CIC is disturbed during processing due to noise, radiation particles, or due to changing *cic_int* or *cic_shift*, then the CIC generates wideband white noise in the output unless flushed. This property is inherent in the mathematics of a CIC filter used for interpolation. The chip incorporates autoflush capability[2] that detects CIC instability and forces the CIC output (and internal states) to zero (typically within 14 clocks of an overflow). The zero duration is 21 clocks, then the CIC filter response ramps up smoothly. A sticky status bit is set (*cic_fl_status*) if this condition occurs. It is cleared by the user writing a zero to the bit. The status bit is normally set by the chip during initialization. It should be cleared by the user after the CIC is active to begin monitoring overflow.

If the signal gain is too high, the signal can overflow internally, which also causes CIC instability. A momentary high power noise spike is seen on the output before the autoflush forces the CIC to zero. If the gain is set so high that the signal rapidly overflows internally, the output appears as a pulsing signal as the CIC periodically overflows.

The CIC may be bypassed by setting *bypass_cic* or by setting *cic_bypass* bit, and setting *cic_int* = 1. This is appropriate for small overall interpolation (< 6) where the CIC filter requirement for wide transition bandwidth would be a problem or when using the chip just for filtering operations.

CIC in Double Rate Mode

In the most common configuration, one PFIR block feeds complex data to a pair of CIC interpolators. The CIC interpolators can also be configured to calculate two results with each clock. In this configuration, the CIC interpolation must be even and the data fed from a filter must be real. Two channels are used in tandem so that one processes the I data and the second processes the Q data. This also provides twice the filtering resource since the filter only needs to process either real or imaginary data, not both. The software configures this if the variable *toutf_rate* is 2. Otherwise, the user must set *cic_2x*, *cic_xmt_d6stg*, *ncic*, and set the mixer to accept the cross-strapped inputs and program the filters to be in *splitiq* mode.

[2] The autoflush mode is a patented feature of the GC5016. Use of the autoflush mode is highly recommended. CIC instability in cellular base stations digital processors without the autoflush feature can cause full power white noise to be transmitted on all frequencies, interfering with cell users in all nearby cells.

Numerically Controlled Oscillator (NCO) and Mixer

The same NCO is in transmit as was described earlier in receive. The mixer can be configured in normal mode or double rate mode. Figure 22 shows the normal mode where each FIR processes a complex data stream and feeds it to a dual CIC block. The dual CIC block is configured to process two streams (I and Q) at the clock rate. The interpolated output is sent onto the mixer.

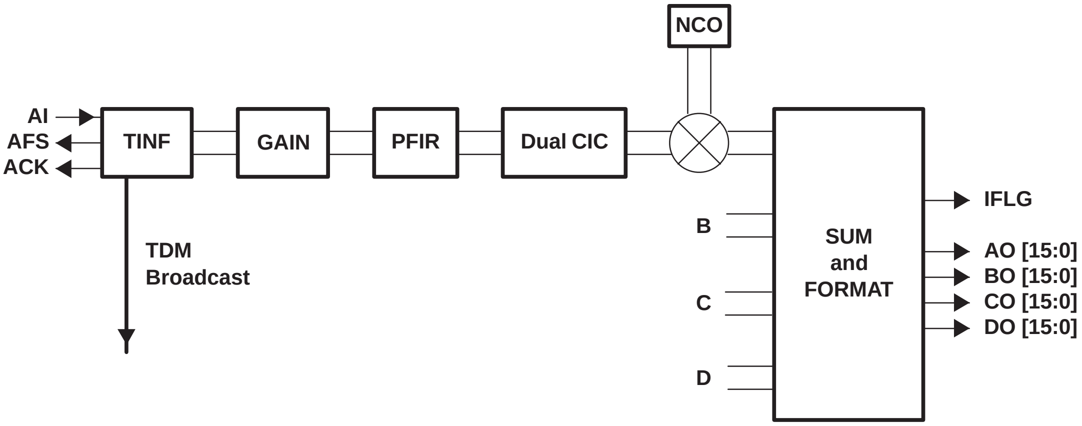


Figure 22. Normal Mode Transmit Channel

Double Rate Mode

Figure 23 shows the double rate data flows from input to mixer output. The example uses channels A and B but channels C and D could be equally used. The real portion of the data must be entered to channel A and the imaginary to channel B. TDM input can be used if desired. The filters A and B are configured to process real data. Each dual CIC is configured to accept real input and interpolate up to two samples per clock. The output of cicA is then I(2k) and I(2k+1), while the output of cicB is Q(2k) and Q(2k+1). Mixer A is configured to accept cross-strapped input for the qcos and qsin multipliers so that mixer A input is I(2k) and Q(2k). Similarly, mixer B is configured to accept cross-strapped input for the icos and isin multipliers, so that mixer B input is I(2k+1) and Q(2k+1). The output sample rate (fsample) is twice the chip clock rate (CK). Each clock the accumulated value in the NCO should be increased by 2 x ftune / fsample x 2^{32} or ftune / CK x 2^{32}. This is set directly in *freq_msb*, *freq_mid*, and *freq_lsb* or by setting *freq* and *fck* in the software. The same value must be programmed for NCO A and B. The *phase* setting for NCO B should be offset from A by one frequency step, so phase(NCO_B) = phase(NCO_A) + ftune / fsample x 2^{16}. The frequency step and accumulated value for NCO A and B are programmed to be the same. NCO B gets a phase offset of ftune / fsample x 2^{16}. When directly programming the chip, these things must be set directly in the control registers. When using the configuration software simply set *tout_rate* to 2.

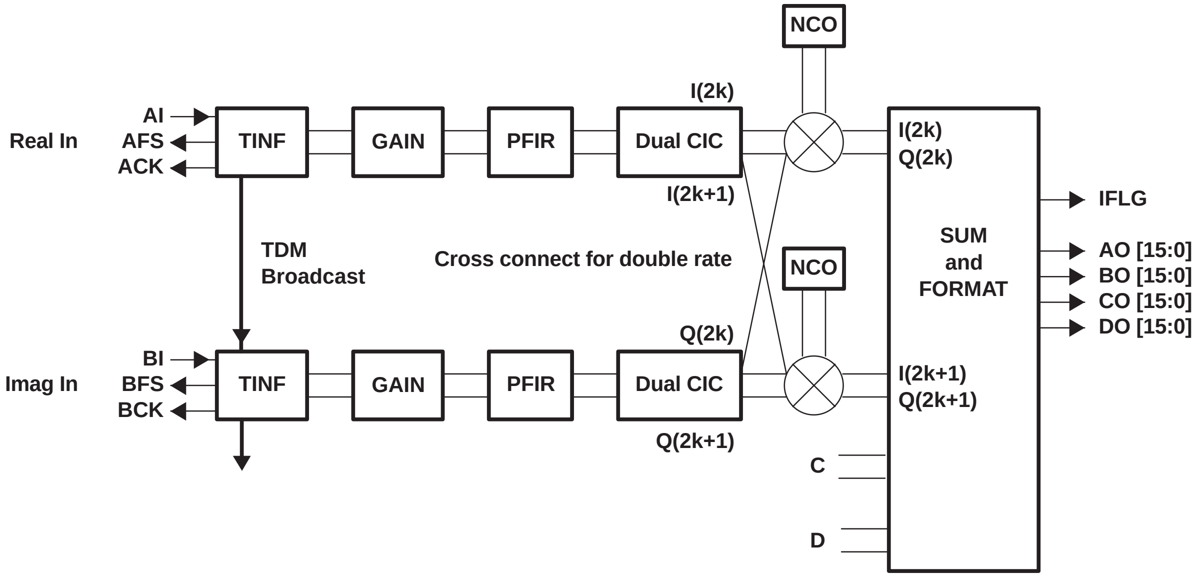


Figure 23. Double Rate Mode Transmit Channel

Transmit Output Buffer

The complex mixer outputs are rounded to 21 bits and sent to the sum tree and transmit output formatter. The GC5016 has a variety of configurations for output of the real or complex up-conversion channel data. There are four 16-bit output ports available in up-conversion mode: AO, BO, CO, and DO. The data pins for the output ports are AO[15..0], BO[15..0], CO[15..0], and DO[15..0].

The rounded 21-bit mixer outputs can either be sent to separate output ports or summed into one or two output signals in a sum tree. The summed signal can also be added to data from an external source such as other GC5016 chips. In this case, ports CO and DO function as sum input ports and are not available signal output. The sum input path and sum output path are expected to be configured the same except for possible rounding in the final chip in a summing chain.

There are five possible modes of operation for the sum tree when CO and DO are used for the sum input path, which are listed in Table 3.

Table 3. Sum I/O Formats

CASE	OUTPUTS	SAMPLE RATE	RESOLUTION	FORMAT	COMMENTS
1	1	CK	22	Real	
2	1	CK/2	22	Complex	I, Q multiplexed
3	1	2 x CK	16	Real	Even and odd demultiplexed
4	1	CK	16	Complex	I, Q parallel
5	2	CK	16	Real	

Case 1: If the data is output as a wide word (22 bits) and at the CK rate, then no time multiplexing is possible and two 16-bit ports are consumed to get the data out. Hence, only real output can be supported.

Case 2: If the output is at CK/2, then I and Q can be time multiplexed allowing one path of wide data, complex at CK/2.

Case 3: If the output is output at 2 x CK, then the data stream must be real and is demultiplexed into even and odd time samples using two output ports.

Case 4: If the output is at CK, then I and Q can be output simultaneously with normal width data using two output ports.

Case 5: If two outputs are used, then only real normal width data can be output using two output ports.

22-Bit SumIO

The sum tree and transmit output formatter for one output is shown in Figure 25. The real or complex mixer outputs are rounded to 21-bits. This includes a 1 bit growth as the partial products of the complex multiply are added together (effectively a gain relative to the msb of 0.5). The sum tree adds up to four up-conversion channels together producing a 23-bit output (and a gain for any channel of 1/4). The 23-bit sum tree output is shifted down by four bits and rounded to 19 bits before being added into the LSBs of the external 22-bit sum input. The gain through the mixer, sumtree, and sumIO is equal to 2^{-7}. When sumIO is bypassed (no sum-in) or is in 16-bit mode, there is no shift down by the four bits, so the gain through the mixer, sumtree, and sumio are equal to 2^{-3}.

The final 22-bit sum is scaled up by 0 to 7 bits (*sum_shift*), checked for overflow, and then sent on to the transmit output formatter. After scaling, the output data is rounded. Overflows in the sum tree are saturated to plus or minus full scale. Hard limiting occurs after shifting and rounding. In normal applications, the data would be rounded to the size of the D/A. The uppermost bits are active and the lower bits are cleared. For applications using sum inputs from other GC5016 chips, rounding should be to 22 bits for all chips except the GC5016 just prior to the D/A. The latency from the sum input port CO and DO to the output ports AO and BO is 14 clock cycles.

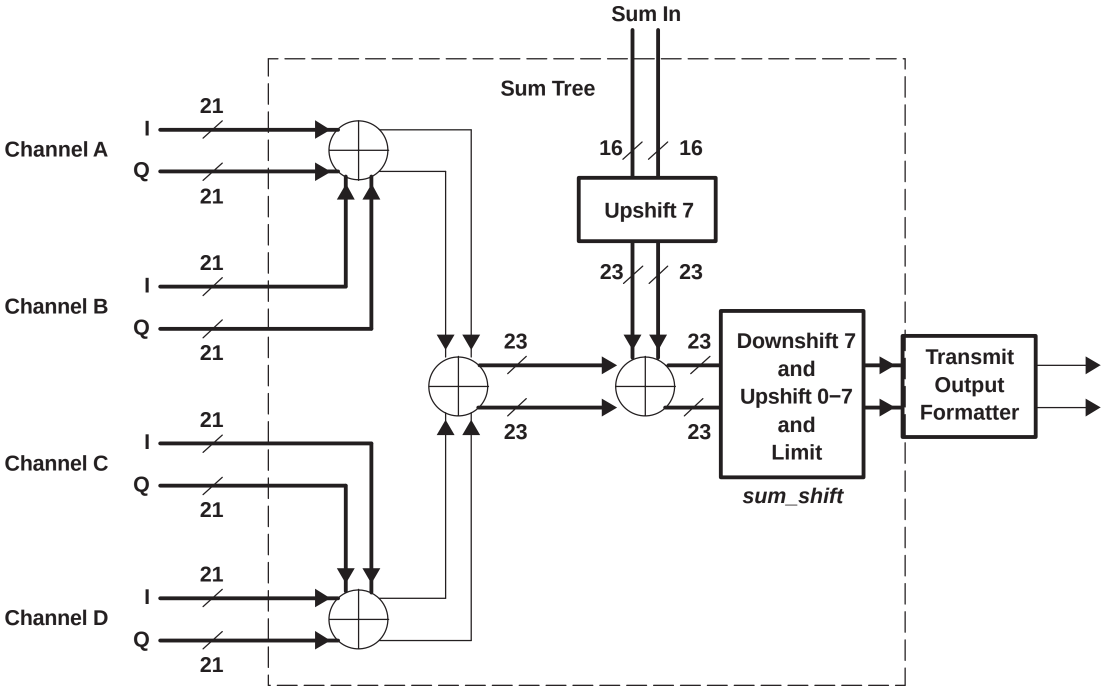


Figure 24. Sum Tree and Transmit Output Formatter − 16-Bit SumIO Mode

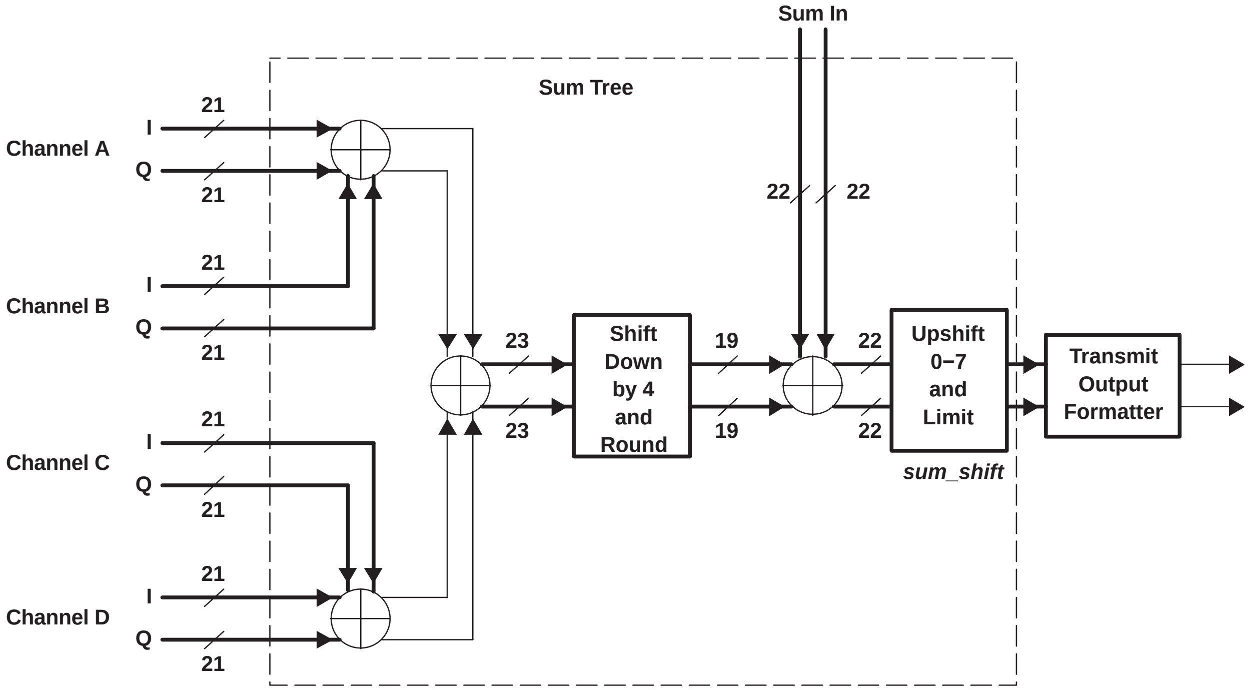


Figure 25. Sum Tree and Transmit Output Formatter − 22-Bit SumIO Mode

Real signals may either be output at full rate, allowing up to four real signals to be output at a time; or real signals can operate at double rate with the data deinterleaved using two ports, with even samples on one port and odd sample on a second port. Wide outputs (>16 bits) require the use of two ports for full-rate output and four ports for double-rate output.

Complex signals may operate at a complex sample equal to half the clock rate (I and Q multiplexed) for four outputs. For one or two complex summed outputs, normal width (< 16 bit) I and Q are output simultaneously on separate ports or wide width (> 16 bits) complex data is output at half the clock rate (I and Q multiplexed). Finally, 16-bit double rate complex data can be output by deinterleaving I and Q onto four separate ports: even I, odd I, even Q, and odd Q.

The configuration program uses the variables *toutf_cmplx* (0 for real and 1 for complex), *toutf_rate* (0 for I/Q multiplexed, 1 for full rate, and 2 for double rate), and *toutf_res* (0 for 16 bits or less and 1 for more than 16 bits) and calculates the appropriate settings for the various hardware parameters.

Sum and Format Details

Figure 26 shows the sum and format blocks in more detail. Normally the configuration software is used to set all these parameters. More details of the implementation are provided here for special applications and to inform interested users.

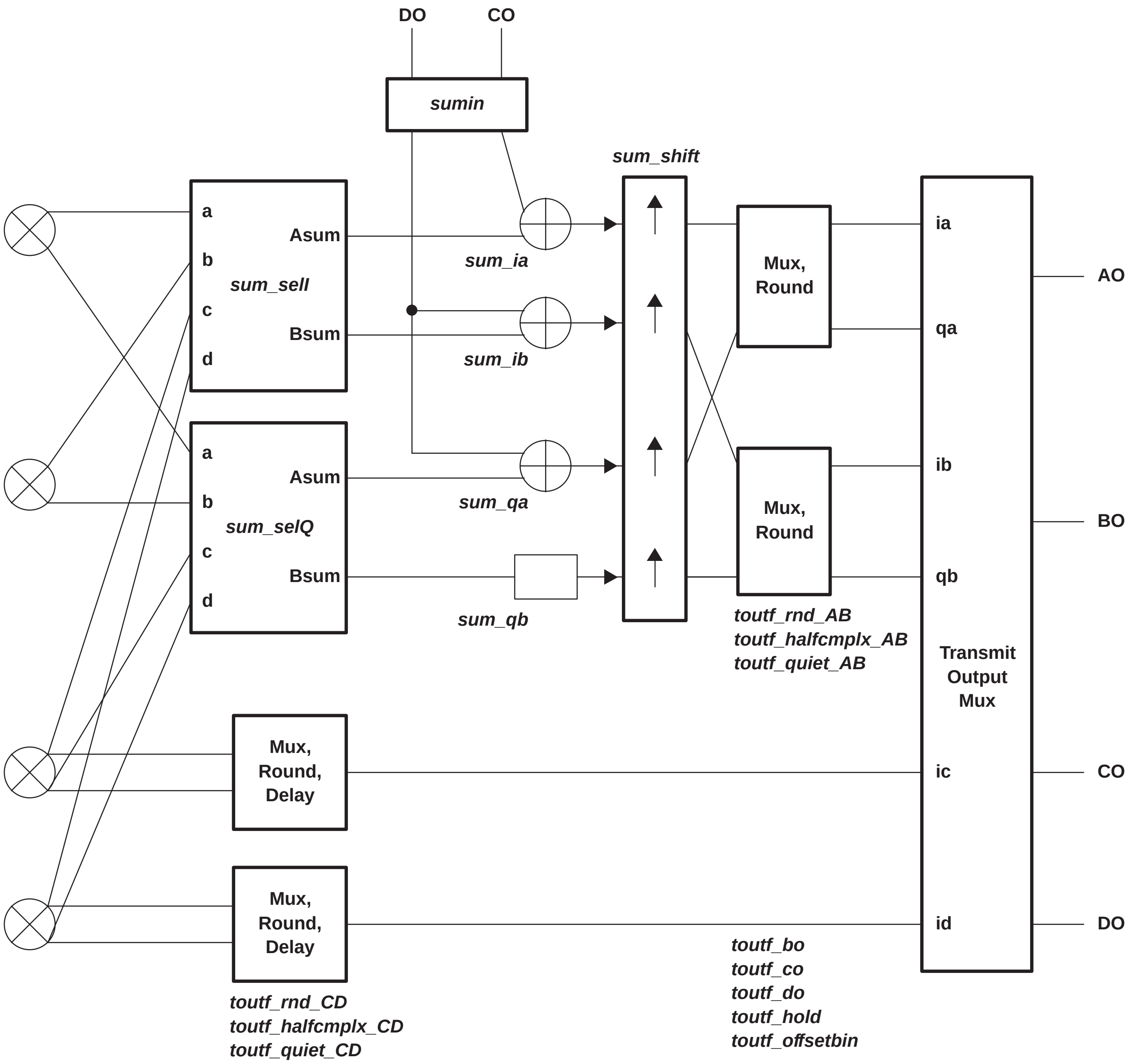


Figure 26. Transmit Sum and Output Format Details

Sum Selection

Data from the four complex mixers (mixA−D) is sent to sum and selection blocks. The I data from the four mixers is combined (controlled by *sum_selI*) into two outputs, Asum and Bsum. Likewise, Q data is combined (controlled by sum_selQ). Sum select controls are an 8-bit value (s7..0) that controls the creation of the two outputs. Asum is s0 x (s1 x a + s2 x b + s3 x c) + s4 x (s5 x d + s6 x b + s7 x c). Bsum is s5 x d + s6 x b + s7 x c. Table 4 shows four common configurations. There is a constraint that s2 and s3 may not both be one. Likewise for s6 and s7.

Table 4. Sum Selection Settings

Sum_sel setting	Asum	Bsum
0x0	0	0
0x43	a	b
0xa7	a+b	c+d
0xb7	a+b+c+d	c+d

Sumin Port for Cascading Chips

If sumin is active, the ports CO and DO are used as inputs (reducing the number of outputs available). The control sumin determines the format of the sum in port data. Both outputs are forced to zero when sumin=0.

When sumin=1, a 22-bit half-rate complex sumin path is formed. The 22 bits are mapped to CO and DO as sumin[21..6] => CO[15..0] and sumin[5..0] => DO[15..10]. The I word is identified by the cic sync. Iflag is high when the I word is expected as an input. The sumin-to-sumout delay is 14 clocks, so when Iflag is high the I word is being output. Summers ib and qb would be programmed off since only one path is available.

When sumin=2, CO[15..0] is passed as a 16-bit value to the sum node controlled by *sum_ia*. DO[15..0] is passed to both *sum_ib* and *sum_qa*. This format is useful for using a sumin path with double rate real, full-rate complex, or two full rate real channels. Due to the limitations of the 16-bit sumpath, gain and SNR need to be carefully analyzed to see if they satisfy the system requirements. In the case of double rate, real summers qa and qb would be programmed off, CO would add to ia, and DO would add to ib. CO should contain I(2k), while DO contains I(2k+1). For full-rate complex summers, ib and qb would be programmed off, CO would add to ia, and DO would add to qa. Finally, for two channels of full-rate real summers, qa and qb would be programmed off, while CO would add to ia and DO would add to ib.

Finally, when sumin=3, a 22-bit full rate sumin path is formed (most common application using sumin). In this case summers ib, qa, and qb are all off. Gain is identical to the 22-bit half-rate complex case discussed above.

If an application calls for both a sumin port and transmit output hold, then the output hold should only be applied to the last chip in a chain.

When cascading chips, *sum_shift* should be set only on the last chip in the chain, it should be 0 in the others. When using a 22-bit sum chain, rounding should be set to 22 bits in all chips except the final one, where the rounding is set appropriate to the next stage in processing (typically a DAC). Likewise, when using a 16-bit sum chain, rounding should be set to 16 bits in all chips except the final one.

Sum Shift

The four paths (ia, qa, ib, and qb) are then upshifted by 0−7 (*sum_shift*). For most applications, different channels added together are independent and their powers should be added in a root-sum-square manner. For optimal performance, gains should be optimized into the D/A, since that is the dominant source of noise. Once the desired level into the DAC is found, gains for the rest of the signal processing chain can be derived. The hardwired gains through the mixer, sumtree, and sumio are set to allow maximal signal growth. The sum shifter allows adjustment in gain for other situations. The mixer allows 1 bit of growth (when one adds real x sin + imag x cos). The 2-bit growth in the sum tree allows for four channels to be added together inside the chip. The 3-bit growth in the sumio (22-bit mode) allows up to 8 GC5016 chips to be cascaded without any clipping prior to the final output shift and round.

With a 16-bit sumio path we can not afford to be so generous with bit growth. Hence, gains with a 16-bit data path are 1 bit in the mixer and 2 bits in the sum tree. This allows a total growth of 3 bits (maximum of four channels for absolutely no clipping or 64 users using RMS) – the user is still limited by signal power per channel in the DAC.

Note that there is no shifter on paths ic and id. These paths are used only when there is no summing, no *sum_in*, and four channels are output (or two channels at double rate). The gain on paths ia and ib can be forced to match ic and id by setting *sum_shift* to 3.

Output Rounding and I/Q Multiplexing

The data on paths A and B next enter mux and round. Multiplexing allows I and Q to be time multiplexed onto the same set of pins. Effectively, this decimates the signal by two and the sync source for this decimation is the *cic_sync*. Finally, *toutf_rndAB* controls rounding of bits from the bottom, so the resulting word is 12 (3), 14 (2), 16 (1), or 22 (0) bits. Bits below the round point are forced to zero. After rounding, the data is hard limited. Paths ia, qa, ib, and qb are 22 bits.

Processing on paths C and D is similar. If channels are added together, channels C and D are routed up and added into paths A and B. If all four channels are output separately then the mux, round, and delay blocks on paths C and D are used. The C and D data is delayed to match hardware pipelining delay on paths A and B. Since there is no summing gain, a fixed upshift of one bit is provided to compensate for the one bit of headroom in the mixer output. The blocks can either output the real data or IQ multiplexed (*toutf_halfcmplx_CD*=1). Paths ic and id out of these blocks are limited to 16 bits, since that matches the size of the output ports. Control *toutf_rndCD* determines the round. Since the path is limited to 16 bits, both *toutf_rndCD*=0 and *toutf_rndCD*=1 round to 16 bits. If paths C and D are unused here (for example when adding paths C and D into paths A and B), then setting *toutf_quiet_CD* forces an overflow condition, thus holding the bus constant (and quiet).

Transmit Output Multiplexing

The final block can invert the msb (*toutf_offsetbin*) for systems that prefer offset binary output to two’s complement.

Port AO always gets ia.

Port BO (*toutf_bo*) can get zeros (0), ib (1), qa (2), the lsb’s of ia (4), or the complement of a(8). Setting 4 is used to support output word sizes greater than 16 bits. If 22 bits are output, then AO gets bits 21..6, while BO[15..10] contain bits 5..0. The remainder of BO is zero. Finally, setting 8 sets BO to be the complement of AO. This can be used together with external resistor packs to create LVDS outputs. Other settings for *toutf_bo* are undefined.

Port CO (*toutf_co*) gets zeros (0), ic (1), ib (2), or qa (4). Other settings for *toutf_co* are undefined.

Port DO (*toutf_do*) gets zeros (0), id (1), qb (2), ib lsb’s (4), qa lsb’s (8), or CO complement (16). Other settings for *toutf_do* are undefined.

Overall Gain in Transmit Mode

The overall gain of the chip is a function of the input gain setting (G), the sum of the programmable filter coefficients, the filter gain (*fir_shift*), the amount of interpolation in the CIC filters (*cic_int*), the scale circuit settings in the CIC filter (*cic_shift*), and the sum tree scale factor (*sum_shift*). The overall gain (22-bit sumio mode) is:

$$\text{GAIN} = \left\{ \left(\frac{\text{G}}{4096}\right)\left(\frac{\text{PFIR\_SUM}}{\mathit{fir\_int} \times 2^{21-\mathit{fir\_shift}}}\right) \mathit{cic\_int}^{(5-\mathit{cic\_xmt\_5stg})}\, 2^{\mathit{cic\_shift}-41} \right\} 2^{\mathit{sum\_shift}-6}$$

where *cic_int*, G, PFIR_SUM, and *fir_shift* can be different for each channel, but *sum_shift* is common to all channels. The term inside { } should be less than or equal to one. For no sumio or 16 bit sumio modes, the gain is:

$$\text{GAIN} = \left\{ \left(\frac{\text{G}}{4096}\right)\left(\frac{\text{PFIR\_SUM}}{\mathit{fir\_int} \times 2^{21-\mathit{fir\_shift}}}\right) \mathit{cic\_int}^{(5-\mathit{cic\_xmt\_5stg})}\, 2^{\mathit{cic\_shift}-41} \right\} 2^{\mathit{sum\_shift}-3} \quad \text{For Channels A and B}$$

$$\text{GAIN} = \left\{ \left(\frac{\text{G}}{4096}\right)\left(\frac{\text{PFIR\_SUM}}{\mathit{fir\_int} \times 2^{21-\mathit{fir\_shift}}}\right) \mathit{cic\_int}^{(5-\mathit{cic\_xmt\_5stg})}\, 2^{\mathit{cic\_shift}-41} \right\} \quad \text{For Channels C and D}$$

GC5016 IN TRANSCEIVER MODE

The GC5016 can also be configured in a transceiver mode as shown in Figure 3, where channels A and B function in up-conversion mode and channels C and D function in down-conversion mode. The channel details for A and B are identical to those described above for up-conversion mode and for C and D are identical to those above for down-conversion mode. Two input ports and two output ports are available each for the up-conversion channels and down-conversion channels. The input and output interfaces are identical to those described above for the up-conversion and down-conversion modes. Sumin is not available in transceiver mode.

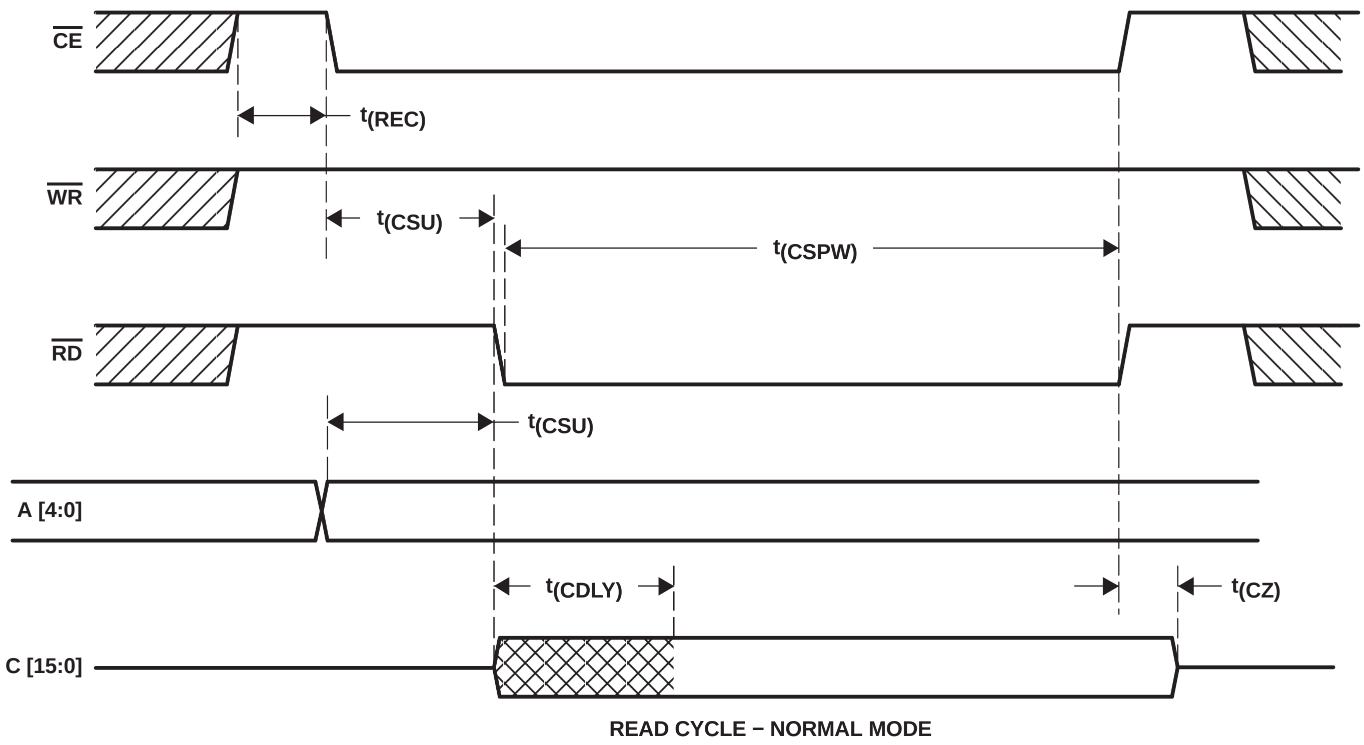
GENERAL GC5016 FEATURES

Control Interface

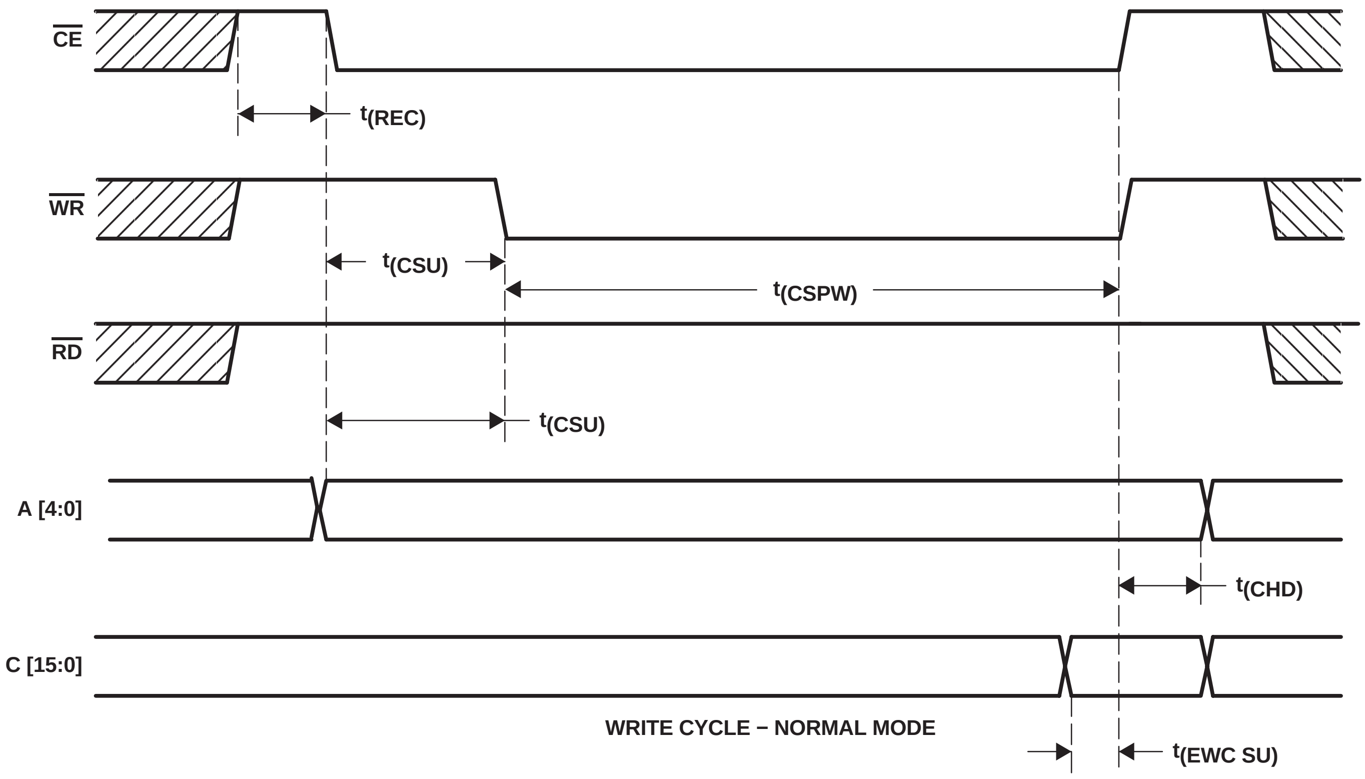
Writing control information into control registers configures the GC5016. The control registers are grouped into eight global registers and 88 pages of registers, each page containing up to 16 registers. The global registers are accessed as addresses 0 through 0xf. Address 0x2 is the page register, which selects which page is accessed by addresses 0x10 through 0x1f. The contents of these control registers and how to use them are described later in this data sheet.

The registers are written to or read from using the C[15..0], A[4..0], $\overline{\text{CE}}$, $\overline{\text{RD}}$, and $\overline{\text{WR}}$ pins. Each control register has been assigned a unique address within the chip. This interface is designed to allow the GC5016 chip to appear to an external processor as a memory mapped peripheral (the pin $\overline{\text{RD}}$ is equivalent to a memory chip's OE pin). The chip supports both dual strobe ($\overline{\text{WR}}$ and $\overline{\text{RD}}$) with a chip select ($\overline{\text{CE}}$) and single strobe ($\overline{\text{CE}}$) with a read/write ($\overline{\text{WR}}$) control. Write timing can either be: 1) edge based where the data bus must be stable for a setup time before and a hold time after ($\overline{\text{CE}}$ **|** $\overline{\text{WR}}$) goes high or 2) latch based where the data must be stable for a setup time before ($\overline{\text{CE}}$ and $\overline{\text{WR}}$) goes low and a hold time after ($\overline{\text{CE}}$ **|** $\overline{\text{WR}}$) goes high. There are four modes supported, with most testing done using edge with dual strobe.

The primary mode is edge timing based (WRMODE=0) with $\overline{\text{RD}}$ active and timing controlled by $\overline{\text{RD}}$/$\overline{\text{WR}}$. In this mode, an external processor (a microprocessor, computer, or DSP chip) can write into a register by setting A[4..0] to the desired register address, setting $\overline{\text{RD}}$ high, selecting the chip by setting $\overline{\text{CE}}$ low, then strobing $\overline{\text{WR}}$ low. The write cycle is active, while both $\overline{\text{CE}}$ and $\overline{\text{WR}}$ are low. Data on the C[15..0] is registered into the chip on the rising edge of $\overline{\text{WR}}$. This is useful for processors that do not assure valid data when the write strobe goes active, but assure that the data is stable for the required setup time before the write strobe goes inactive. To read from a control register the processor must set A[0:4] to the desired address, select the chip with the $\overline{\text{CE}}$ pin, and then set $\overline{\text{RD}}$ low. The chip then drives C[0:15] with the contents of the selected register. After the processor has read the value from C[0:15] it should set $\overline{\text{RD}}$ and $\overline{\text{CE}}$ high. The C[0:15] pins are turned off (high impedance) whenever $\overline{\text{CE}}$ or $\overline{\text{RD}}$ are high or when $\overline{\text{WR}}$ is low. The chip only drives these pins when both $\overline{\text{CE}}$ and $\overline{\text{RD}}$ are low and $\overline{\text{WR}}$ is high.



READ CYCLE − NORMAL MODE



WRITE CYCLE − NORMAL MODE

Figure 27. Control Timing for Normal Mode (Edge with Dual Strobe)

Some processors provide a single control $\overline{RD}/\overline{WR}$ together with a chip strobe that controls timing. In this case, the $\overline{RD}$ pin can be grounded. The control processor must set A[4..0] to the desired register address, set $\overline{WR}$ low for a write or high for a read, and select the chip by setting $\overline{CE}$ low. The write cycle is active while both $\overline{CE}$ and $\overline{WR}$ are low. Data on the C[15..0] is registered into the chip on the rising edge of $\overline{CE}$.

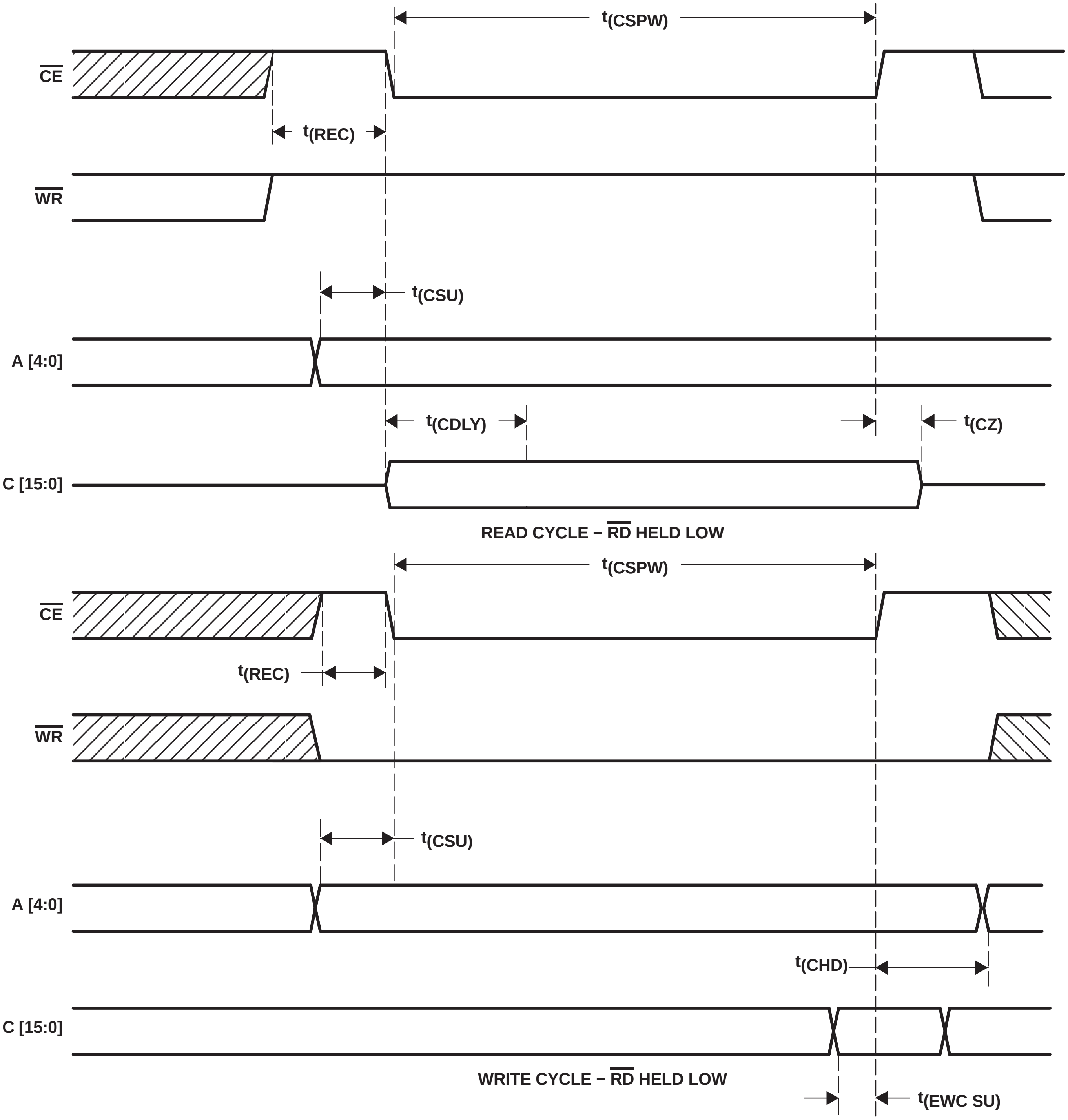
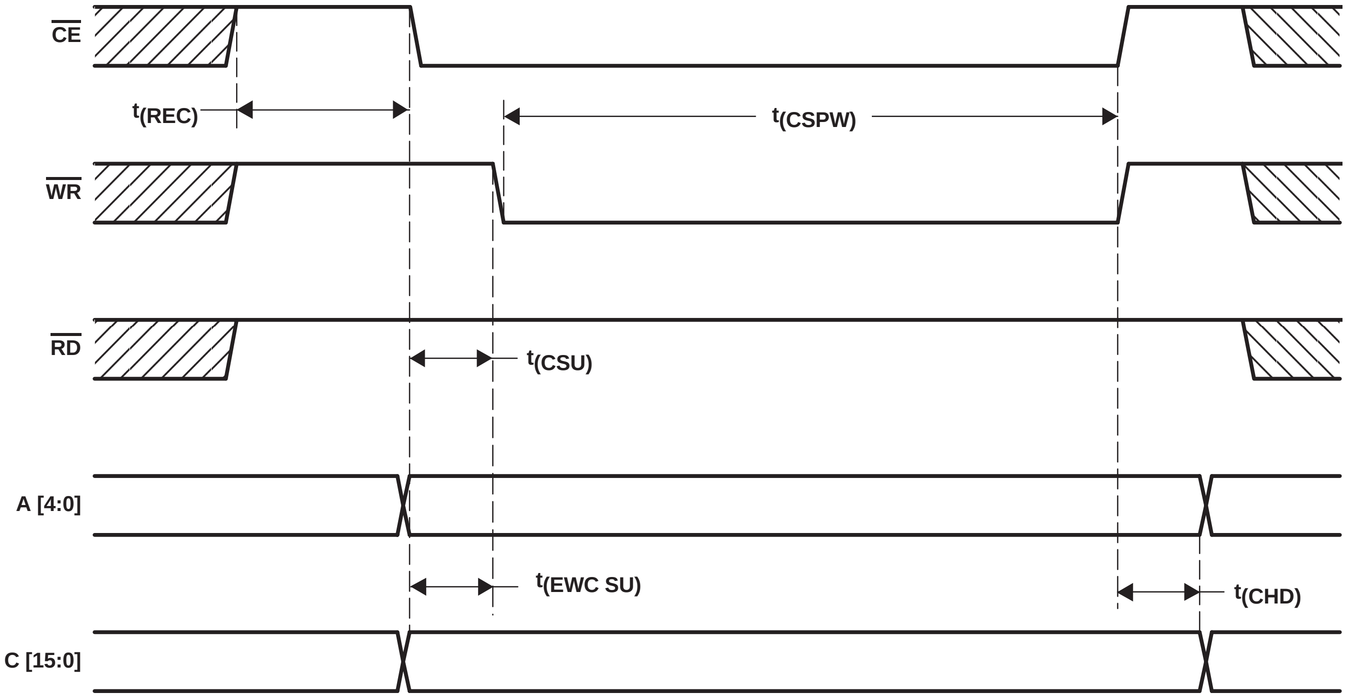


Figure 28. Control Timing for Edge With Single Strobe

Some processors assure data is stable from a setup time ahead of $\overline{\text{WR}}$ going low to a hold time after $\overline{\text{WR}}$ goes high. For these processors, latch mode (WRMODE=1) can be used. In this mode, the data on C[15..0] is transferred to the control registers during the entire time both $\overline{\text{WR}}$ and $\overline{\text{CE}}$ are low. Since some control registers (such as most sync registers) are sensitive to transient values on the C[0:15] data bus, the data must be stable during the entire write pulse in this mode. If the processor controls timing using $\overline{\text{RD}}$ and $\overline{\text{WR}}$ the following timing diagram is appropriate for writes. For reads, the timing is identical to the edge with dual strobes shown in Figure 27.



WRITE CYCLE − LATCH MODE

Figure 29. Control Timing for Latch With Dual Strobe

Finally, the user can also use the latch mode with a single strobe, as shown in Figure 30. Timing for reads is the same as edge with the single strobe shown in Figure 28.

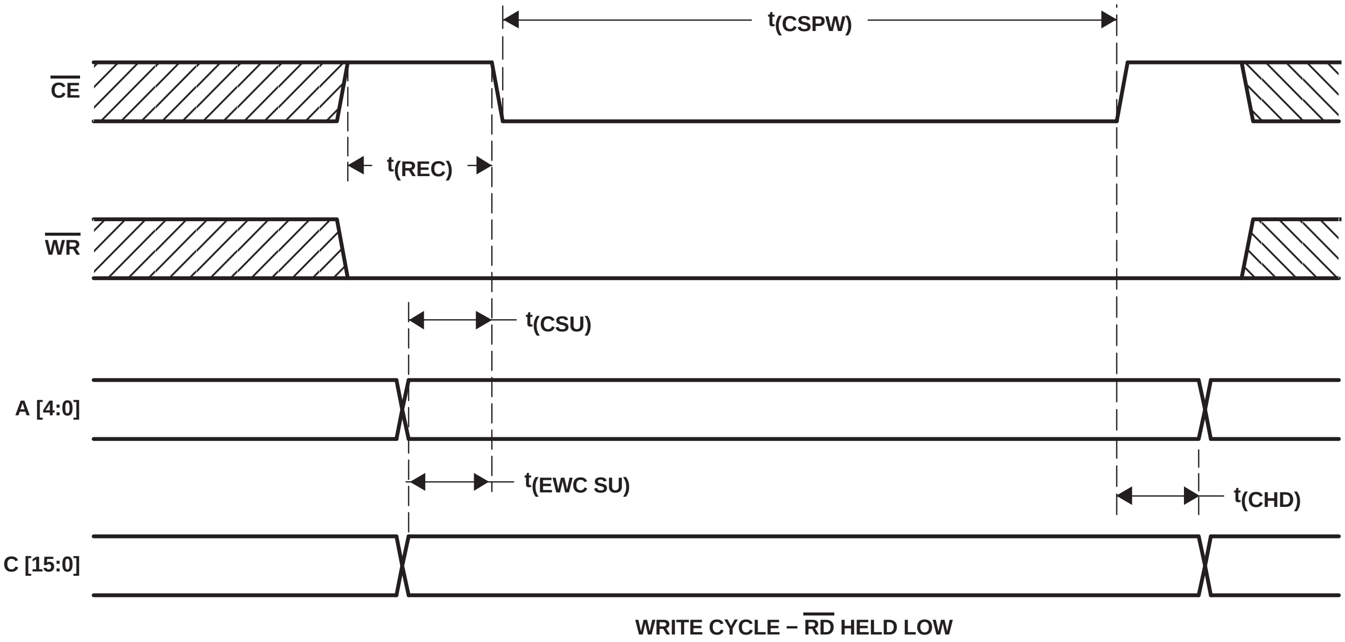


Figure 30. Control Timing for Latch With Single Strobe

Clocking

The GC5016 uses a simple clocking scheme. The same clock is used throughout the chip (gated in some areas to save power). Note that *rinf_zpad* can be used to allow the input sample rate (in receive) to be a fraction of the chip clock rate. Likewise, *toutf_hold* can be used to allow the output sample rate (in transmit) to be a fraction of the chip clock rate. All channels use the same clock, even in a transceive configuration.

Power-Down Modes

The GC5016 allows software control to power down each of the four channel filters and each of the two cic/mix blocks. When a block is powered down, control registers are not altered. The state machines and data paths are put into a reset state.

Synchronization

Each GC5016 chip can be synchronized through the use of one of two sync input signals, an internal one shot sync generator, or a sync counter. The sync to each circuit can also be set to be always on or always off. Each circuit within the chip, such as the sine/cosine generators or the decimation control counter can be synchronized to one of these sources. These syncs can also be output from the chip so that multiple chips can be synchronized to the syncs coming from a designated master GC5016 chip.

The 3-bit sync mode control for each sync circuit is defined in Table 5.

Table 5. Sync Modes

MODE	SYNC SOURCE
0, 1	Off (never asserted)
2	SIA
3	SIB
4	*one_shot*
5	TC (terminal count of internal counter)
6, 7	On (always active)

NOTE: The internal syncs are active high. The $\overline{\text{SIA}}$ and $\overline{\text{SIB}}$ inputs have been inverted to be the active high syncs SIA and SIB in Table 5.

The *one_shot* can either be a level or a pulse as described in Table 15. The level mode is used to initialize the chip; the pulse mode is used to synchronously switch frequency, phase, or gain values.

Typically the decimation counters (*dec_sync*), the flush circuits (*flush_sync*), and the output block sync (*out_blk_sync*) is set to SIA, while the NCO phase accumulator syncs (*nco_sync*) is set to SIB. The $\overline{\text{SIA}}$ input can then be used to initialize and flush the channels and the $\overline{\text{SIB}}$ sync input can be used, if desired, to synchronize the phases of the NCOs.

The recommended sync mode settings are summarized in Table 6.

The $\overline{\text{SIA}}$ and $\overline{\text{SIB}}$ sync inputs are either connected to a user defined sync generator, for example, an FPGA, or are tied to a GC5016 chip's sync output pin ($\overline{\text{SO}}$). If there are multiple GC5016 chips in the system, then the $\overline{\text{SO}}$ pin of one chip can be used to drive the $\overline{\text{SIA}}$ input of all chips, and the $\overline{\text{SO}}$ pin of another chip can drive the $\overline{\text{SIB}}$ inputs of all chips.

Table 6. Recommended Sync Settings

Global Syncs (Address 1)			**Channel Syncs (Pages 0x14, 0x34, 0x54, and 0x74 Address 0x15)**		
Sync	**Value**	**Description**	**Sync**	**Value**	**Description**
soB_sync	4 (OS)	The SO output is used during initialization	*fir_sync*	2 ($\overline{\text{SIA}}$)	Sync FIR during initialization
CIC and Mixer Syncs (Page 0x80 and 0xa0, Addresses 0x16 and 0x1e)			*coef_sync*	7 (always)	For use with multiple coefficient sets
Sync	**Value**	**Description**	*gain_sync*	4 (OS)	Sync gain when changing base value
freq_sync	7 (always)	Use frequency settings as they are loaded	*power_sync*	7 (always)	Only sync when measuring power
phase_sync	7 (always)	Use phase settings as they are loaded	*sck_sync*	4 (OS)	Sync the serial clock during initialization
dith_sync	0 (never)	Can free run. Set to 7 (always) to disable			
nco_sync	3 ($\overline{\text{SIB}}$)	For NCO updates. Disrupts signal processing when sync occurs.			
flush_sync	0 (never)	Can free run. Set to 5 (counter) for checksum			
cic_sync	2 ($\overline{\text{SIA}}$)	Sync CIC during initialization			

This arrangement allows the user to use the $\overline{\text{SO}}$ sync output to synchronously drive the $\overline{\text{SIA}}$ or $\overline{\text{SIB}}$ sync inputs of all chips. They sync source for $\overline{\text{SO}}$ is selected using the *output_sync* control bits in address 4.

Initialization

Three initialization procedures are recommended. The first is for standalone GC5016 chips, the second is for a multi-GC5016 chip configuration synchronized by a master GC5016 chip, and the third is for a configuration where the GC5016s are to be synchronized by to an external source.

Standalone GC5016 Chips

This procedure works if the GC5016 can free run and its timing doesn't need to be synchronized with other chips (FPGAs, other GC5016s, etc.).

1. Reset the chip by writing 0xFF00 to address 0.
2. Disable all outputs by writing 0 to address 3.
3. Force the one-shot to be a level 1 by writing 0x1C to address 1.
4. Load the configuration generated by the cdm5016 program.

Note: All sync controls should be set to 4, so that they are controlled by the one-shot.

This can be done by adding these lines to the cmd5016 input file:

soB_sync 4

fir_sync 4

dith_sync 4

nco_sync 4

cic_sync 4

5. Clear the reset by writing 0x100 to address 0.
6. Release the syncs by writing 0x14 to address 1.

Multiple GC5016 Chips Using $\overline{\text{SO}}$ From a Master GC5016 Chip

This procedure works if multiple GC5016 chips need to be synchronized. This assumes that the sync output pin ($\overline{\text{SO}}$) from the master GC5016 chip is connected to the $\overline{\text{SIA}}$ input pin on all GC5016 chips including the master chip.

1. Reset the chip by writing 0xFF00 to address 0 of all chips.
2. Disable all outputs except $\overline{\text{SO}}$, by writing 0x100 to address 3 of all chips.
3. Force the one-shot to be a level 1 and use the one shot to drive $\overline{\text{SO}}$ by writing 0x1C to address 1 in all chips.
4. Load the configuration generated by the cdm5016 program to all chips.

Note: All sync controls except for $\overline{\text{SO}}$, should be set to 2 so that they are controlled by the $\overline{\text{SIA}}$ input sync.

This can be done by adding these lines to the cmd5016 input file:

soB_sync 4

fir_sync 2

dith_sync 2

nco_sync 2

cic_sync 2

5. Clear the reset by writing 0x100 to address 0 of all chips.
6. Release the syncs by writing 0x14 to address 1 of the master chip.

Multiple GC5016 Chips Using an External Sync

Same as above, except the $\overline{\text{SIA}}$ input is tied to an FPGA or other sync source.

1. Reset the chip by writing 0xFF00 to address 0 of all chips.
2. Disable all outputs except $\overline{\text{SO}}$, by writing 0x100 to address 3 of all chips.
3. Drive the $\overline{\text{SIA}}$ inputs low (active) using the external sync source.
4. Load the configuration generated by the cdm5016 program to all chips.

Note: All sync controls should be set to 2, so that they are controlled by the $\overline{\text{SIA}}$ input sync.

This can be done by adding these lines to the cmd5016 input file:

soB_sync 2

fir_sync 2

dith_sync 2

nco_sync 2

cic_sync 2

5. Clear the reset by writing 0x100 to address 0 of all chips.
6. Release the syncs by setting $\overline{\text{SIA}}$ high (inactive).

NOTE: $\overline{\text{SIA}}$ is clocked into the GC5016 chips by CK and the signal must meet the specified setup and hold times for all of the GC5016 chips.

Diagnostics

The GC5016 provides self-test capability by providing a pattern generator at the inputs, a checksum (also known as a signature) generator at the outputs, and a general timer. There is a pattern generator at the receiver inputs, which can generate a linear feedback sequence, a constant, or a ramp. There is a separate pattern generator at the transmit inputs, which can generate a linear feedback sequence or a constant. At each of the four receive outputs and at the transmit output there are checksum blocks. The checksum is generated using another LFSR with the chip outputs feeding into the current state. The chip must be set up to synchronize all blocks together with the pattern generator and the checksum. At each sync time the test pattern is repeated, the checksum is transferred from a calculation register to a uP register, and the state of the checksum LFSR is reset. Several self-test configurations are added as an appendix later. The NCO must be sync'd with the data pattern (so the accumulated phase always starts at zero). The dither sync must either be set to always on (thus freezing the dither) or sync'd to the sync source. The user should wait for at least four sync periods to allow the checksum to stabilize before reading the checksum.

JTAG

The GC5016 supports JTAG with a 5-pin interface. The JTAG implementation supports the standard boundary scan (used for board test), and chip identification. A BSDL file is available on the web. The GC5016 identification string is a 1 followed by an 11-bit manufacturer number (0x8c), a 16-bit chip number (5016 = 0x1398), and a 4-bit revision number (currently a 1).

Mask Revision Register

The mask revision register contains a simple 8-bit code that changes with any mask changes which may impact software, so that users can deploy software that tests for which revision and changes the behavior accordingly. The current revision value is 1.

CONFIGURATION SOFTWARE

The cmd5016 is a configuration software program for the GC5016. The program accepts inputs from the user describing the desired signal processing and the program generates a set of output files. These include a configuration file, an analysis file, a debug file, and a table file. It is anticipated that the software is embedded behind a user interface. This description is for the command line version, but should still be helpful for all versions.

The configuration file describes the settings for all registers in a format suitable for the GC101 evaluation module (and likely easily modified for other uses). This file contains a long sequence of register writes to the chip.

The analysis file provides the user with feedback on gains internal to the chip and any other useful information intended for the user to read, rather than for software to use.

The debug file is intended for the authors of the software program and is likely too cryptic for anyone else.

The table file contains the values for each of the control fields. This format is easier for people to see what setting was generated for each control field. The configuration file is more work for people to read, since the fields are merged into registers and the registers must be found by finding the page write, then the register write.

Errors are reported to the screen and to all output files. They must be fixed before the chip is configured. Warnings are sent to the analysis file and indicate unusual but possibly legal conditions.

Each channel has two modes of operation transmit and receive. The chip may operate with all four channels in transmit, all four in receive, or two in transmit (AB) and two in receive (CD). The mode settings must be declared first as different parameters are legal for different modes. The mode is specified using a mode command:

```
mode [AB | CD] [transmit | receive]
```

The user sets variables in the command input file. These variables may be fields (bit field of the hardware control registers) or *pseudo_fields* (variables that exist in software only and have no directly corresponding element in hardware). Variables can be one of six types: mandatory (M), defaulted (D), computed (C), unused (X), expert (E), and not applicable (−). Not applicable applies to variables that are read only active (such as page or one-shot). Variables can have different types in transmit and receive mode. For example, the CIC interpolation (*cic_int*) is mandatory in transmit and unused in receive. Any variable may be directly set by the user and that value is used both to program the chip and in some cases to compute other fields. A user should first let the software generate values for computed and expert fields before attempting to modify them. Setting *expert_only* fields gives a warning and generally should not be attempted. Computed fields (such as *gain_lsb* and *gain_msb*) can reasonably be changed by a user – but most commonly should not need be changed. Defaulted values should be reviewed to see if they fit the user’s application. Mandatory values must be set – the software does not generate a configuration without them. Unused variables are programmed to zero – though it does not matter. Any user set variables are normally used to compute other fields. The override command is used to allow the user to override the settings of specific variables without impact to any other variables. The user’s control file is read up to the override command, all computations and analysis are done and reported, the rest of the control file is read to override specific settings, and finally the control register settings are written.

A comment consists of a # or / at the start of a line. Blank lines are allowed.

Table 7 describes the *pseudo_fields*. The *Rcv* column indicates the type in receive mode. The *Xmt* column indicates the type in transmit mode. The *TYPE* column indicates if the variable is per channel or global. Names in the text in italics refer to cmd5016 variable names described in the following tables.

Table 7. Pseudo Fields in cmd5016

Rcv	Xmt	NAME	TYPE	DEFAULT	DESCRIPTION
M	M	*bits*	channel	0	Bits in each word of the output data in receive or input data in transmit. Must be either 4, 8, 12, 16, or 20. An interface with 8-bit real and 8-bit imaginary data would be programmed to 8.
D	D	*bypass_cic*	channel	0	Programs the CIC into bypass mode (1) or normal (0). It neither interpolates nor decimates. Gain is unity in transmit, 1/2 in receive. The data still goes through the CIC.
D	D	*bypass_fir*	channel	0	Programs the FIR filter to an impulse, with a minimum latency and unity gain (1) or normal.
D	D	*bypass_mix*	channel	0	Programs the mixers to be nearly unity gain (2^20−1)/2^20. The sin mixer multipliers are programmed to multiply by zero. Also sets the frequency to be dc and phase to zero.
M	X	*cic_dec*	channel	1	CIC decimation amount (only in receive).
X	M	*cic_int*	channel	1	CIC interpolation amount (only in transmit).
M	X	*fir_dec*	channel	1	FIR decimation (only in receive). Values from 1 to 16.
D	D	*fir_diff*	channel	0	If set, allows multiple data streams in the filter to get different coefficients.
X	M	*fir_int*	channel	1	FIR interpolation amount (only in transmit) (1−16).
D	D	*fir_nchan*	channel	1	Specifies how many data streams in the filter (1−16). Almost always the default of 1 is correct.
M	M	*pins*	channel	0	Specifies how many pins are active in the output interface in receive or input interface in transmit. Must be either 4, 8, or 16.
D	X	*rin_cmplx*	global	0	Receive input data is complex (1) or real (0).
D	X	*rin_rate*	global	1	Receive input data rate is half (0) for IQ time multiplexed, full (1), or double (2). If double, *splitiq* must also be set.
D	X	*routf_tdm*	global	0	Receive output data is TDM'd (1) onto port DO or normal (0).
D	D	*splitiq*	global	0	If set, complex IQ data is split so the I data goes to one filter and Q data goes to another.
X	M	*tinf_cmplx*	channel	NA	Transmit input data is complex (1) or real (0). If *splitiq* is on *tinf_cmplx* must be real.
X	D	*tinf_fs_dly*	channel	3	Transmit input frame strobe delay. Delays internal strobe identifying the msb from the output frame strobe. 4 bits
X	D	*tinf_tdm*	global	0	TDM'd transmit input data is expected on port AI(1), or normal (0).
X	D	*tout_cmplx*	global	1	Transmit output data is complex (1) or real (0).
X	D	*tout_nsig*	global	NA	Number of transmit output signals. Defines how channels are summed together. The number of input signals is derived from channels in transmit mode that are powered and whether they are in *splitiq* or not.
X	D	*tout_rate*	global	1	Transmit output rate is half (0) for IQ time multiplexed, full (1) or double (2). If double, *splitiq* must also be set.
X	D	*tout_res*	global	1	Transmit output resolution is normal (0) for 16 bits or less, high resolution (1) up to 22 bits, or complementary signaling (2).
X	D	*tout_sumin*	global	0	Sumin port active (1) or not (0).
M	M	*freq*	channel		Frequency. Either *freq_msb* or *freq* must be set. If *freq* is set, it is read as a floating point number and *freq_msb*, *freq_mid*, *freq_lsb* are calculated. If both are set a warning is issued.
M	M	*fck*	global		The chip clock rate. It is used to compute estimated power as well as to convert *freq* to *freq_msb*, *freq_mid*, and *freq_lsb* (including effects of double, *rin_zpad* and *tout_hold*).

Mandatory variables for transmit are *bits*, *cic_int*, *fir_int*, (*freq* OR *freq_msb*), *pins*, and *tinf_cmplx*. Mandatory variables for receive are *bits*, *cic_dec*, *fir_dec*, (*freq* OR *freq_msb*), *pins*, and *mix_rcv_sel*.

Remaining variables are described in the hardware register descriptions below.

Control Registers

This section describes the control registers of the GC5016. The configuration program calculates values for most of the fields. Users are encouraged to first go to the software description, then to look here for more details on particular fields of interest. Five address pins, plus 8 bits of page in the page register, identify which register is being addressed. Addresses, page numbers, and bit values are in hex. Global registers ignore page register content and are in addresses 0−f, while paged registers are in addresses 10−1f.

Table 8 provides an overview of the page allocations. FirA−D, cicAB, and cicCD are hardware blocks that contain the signal processing blocks listed in Table 8.

The register values are not changed by the hardware-reset pin, software master reset, reset to any block, or clock loss. A global register 0 (internal reset and power down) is set at power up (but not by any reset action). Global register 3 (output enables) is cleared at power up.

Table 8. Map of Control Pages

BLOCK	PAGES	DESCRIPTION
FirA	00−1f	Transmit input formatter, gain, PFIR, AGC, power meter, receive output formatter for A
FirB	20−3f	Transmit input formatter, gain, PFIR, AGC, power meter, receive output formatter for B
FirC	40−5f	Transmit input formatter, gain, PFIR, AGC, power meter, receive output formatter for C
FirD	60−7f	Transmit input formatter, gain, PFIR, AGC, power meter, receive output formatter for D
cicAB	80−81	CIC, NCO, mixers for channels A and B, common transmit blocks
cicCD	a0−a1	CIC, NCO, mixers for channels C and D, common receive blocks

Table 9 lists the global registers in the chip.

Table 9. Global Control Registers

PAGE	ADDRESS	REGISTER DESCRIPTION
Global	0	Reset and clock control
Global	1	General sync
Global	2	Page and revision
Global	3	Output enables

Table 10 lists the registers in FirA. Identical registers are present for FirB, FirC, and FirD with page offsets of 20, 40, and 60 respectively.

Table 10. FirA Control RAMs

BLOCK	PAGES	DESCRIPTION
FirA	00−1f	Transmit input formatter, gain, PFIR, AGC, power meter, receive output formatter for A
FirB	20−3f	Transmit input formatter, gain, PFIR, AGC, power meter, receive output formatter for B
FirC	40−5f	Transmit input formatter, gain, PFIR, AGC, power meter, receive output formatter for C
FirD	60−7f	Transmit input formatter, gain, PFIR, AGC, power meter, receive output formatter for D
cicAB	80−81	CIC, NCO, mixers for channels A and B, common transmit blocks
cicCD	a0−a1	CIC, NCO, mixers for channels C and D, common receive blocks

Table 11 lists the control registers for FirA.

Table 11. FirA Control Registers

PAGE	ADDRESS	REGISTER DESCRIPTION
0−f	10−1f	FIR coefficients
10	10−1f	Swap ram contents
11	10−1f	BE ram contents
12	10	Coefficient address generator
12	11	Common address generator
12	12	Forward read address generator
12	13	Forward write address generator
12	14	Backward read address generator
12	15	Backward write address generator
12	16	Backward end cell read address generator
12	17	Backward end cell write address generator
12	18	Forward write strobe
12	19	Backward write strobe
12	1a	Backward end cell read bypass
13	10	Transmit input formatter
13	11	Transmit frame strobe controls
13	12	Transmit frame counter
13	13	Gain 16 lsb’s
13	14	Gain controls
13	15	AGC minimum adaptation limit
13	16	AGC maximum adaptation limit
13	17	AGC counts and threshold
13	18	AGC loop gains
13	19	AGC gain read back
13	1a	Power meter least significant 16 bits
13	1b	Power meter most significant 16 bits
13	1c	Power meter status
13	1d	Power meter Integration time
13	1e	Receive output formatter
13	1f	Receive checksum
14	10	FIR swap ram controls
14	11	FIR accumulator controls
14	12	FIR output control
14	13	FIR sync count
14	14	FIR clock
14	15	Channel syncs

Table 12 lists the control registers for cicAB.

Table 12. Control Registers for cicAB

PAGE	ADDRESS	REGISTER DESCRIPTION
80	10	A CIC mode
80	11	A phase
80	12	A frequency (lsb)
80	13	A frequency (mid)
80	14	A frequency (msb)
80	15	A mixer
80	16	A NCO sync and dither control
80	17	A CIC count and sync
80	18	B CIC mode
80	19	B phase
80	1a	B frequency (lsb)
80	1b	B frequency (mid)
80	1c	B frequency (msb)
80	1d	B mixer
80	1e	B NCO sync and dither control
80	1f	B CIC count and sync
81	10	Sum tree sum selection
81	11	Sum tree multiplexing
81	12	Receive sensitivity reduction path A
81	13	Receive sensitivity reduction path B
81	14	Receive sensitivity reduction path C
81	15	Receive sensitivity reduction path D

Table 13 lists the control registers fosr cicCD.

Table 13. Control Registers for cicCD

PAGE	ADDRESS	REGISTER DESCRIPTION
a0	10	C CIC mode
a0	11	C phase
a0	12	C frequency (lsb)
a0	13	C frequency (mid)
a0	14	C frequency (msb)
a0	15	C mixer
a0	16	C NCO sync and dither control
a0	17	C CIC count and sync
a0	18	D CIC mode
a0	19	D phase
a0	1a	D frequency (lsb)
a0	1b	D frequency (mid)
a0	1c	D frequency (msb)
a0	1d	D mixer
a0	1e	D NCO sync and dither control
a0	1f	D CIC count and sync
a1	10	Receive input formatter
a1	11	General timer
a1	12	Receive syncs
a1	13	Transmit output multiplexing
a1	14	Transmit output rounding and hold
a1	15	Transmit checksum

Global Registers

The following tables describe the various bit fields contained in each of the global control registers.

Table 14. Global Register Reset and Clock Control Address 0x0 Bits 15..8 Set at Power Up

Rcv	Tx	FIELD	BITS	Dflt	DESCRIPTION
−	−	*ck_loss_status*	1..0		Bit 0 is current status, bit 1 is sticky status. Set if clock loss is detected. The user must reset bit 1.
D	D	*en_ck_loss*	8	1	Enable clock loss detection. Highly recommended.
C	C	*pwr_dwn_fir_A*	9		Power down PFIR in A. Power down puts the block in reset and disables the clock. It does not reset the control registers.
C	C	*pwr_dwn_fir_B*	10		Power down PFIR in B
C	C	*pwr_dwn_fir_C*	11		Power down PFIR in C
C	C	*pwr_dwn_fir_D*	12		Power down PFIR in D
C	C	*pwr_dwn_cic_AB*	13		Power down cic/mix for A and B
C	C	*pwr_dwn_cic_CD*	14		Power down cic/mix for C and D
C	C	*master_reset*	15		Power down entire chip

Table 15. General Sync Global Address 0x1

RCV	TX	FIELD	BITS	Dflt	DESCRIPTION
D	D	*soB_sync*	2..0	2	Signal selection for *sync_out_B*
−	−	*one_shot*	4..3		One shot control. (=0) Armed issue one shot when lsb goes high; (=1) issues one shot on transition of lsb to high safe state to be left in; (=2) level output 0; (=3) level output 1

Table 16. Page and Revision Global Address 0x2

Rcv	Tx	FIELD	BITS	Dflt	DESCRIPTION
−	−	*page*	7..0		Page register
−	−	*chip_rev*	15..8		Chip revision. Read only. Currently 1.

Table 17. Output Enables Global Address 0x3 Cleared at Power Up

Rcv	Tx	FIELD	BITS	Dflt	DESCRIPTION
C	C	*en_AO*	0		Enable data output AO
C	C	*en_AFS*	1		Enable frame strobe and output clock for A
C	C	*en_BO*	2		Enable data output BO
C	C	*en_BFS*	3		Enable frame strobe and output clock for B
C	C	*en_CO*	4		Enable data output CO
C	C	*en_CFS*	5		Enable frame strobe and output clock for C
C	C	*en_DO*	6		Enable data output DO
C	C	*en_DFS*	7		Enable frame strobe and output clock for D
C	C	*en_soB*	8		Enable *sync_out_B* and iflag
D	D	*ckp_A*	9	0	Invert ACK
D	D	*ckp_B*	10	0	Invert BCK
D	D	*ckp_C*	11	0	Invert CCK
D	D	*ckp_D*	12	0	Invert DCK
X	D	*sumin_clr*	13	0	Force sumin port to zero

FIR Control RAMs

The programmable filter has three RAM’s used to control its operation.

PAGE	ADDRESS	REGISTER DESCRIPTION
0−f	10−1f	FIR coefficients
10	10−1f	Swap ram contents
11	10−1f	BE ram contents

The filter coefficients are stored in a 16-word (by 16 bit) RAM, in each of 16 filter cells. The coefficients are 16-bit two’s complement. The coefficients can be read without interrupting normal operation. Changing the coefficients during normal operation can cause erroneous output should the hardware be reading a coefficient value simultaneously. The coefficients can be divided into banks to allow safe updating and synchronous changing to a new set. The coefficients are stored in addresses 0x10 to 0x1f on pages 0x0 to 0xf (for channel A), pages 0x20 to 0x2f (for channel B) etc. The configuration software takes the filter coefficients from a file and writes them to the appropriate RAM locations.

The swap RAM read address is done using a counter and a 16 word by 4-bit RAM. Normally, this RAM is programmed so the content of each location equals its address (effectively bypassing it). In the future, this RAM can be used to allow complex coefficients by allowing the same data to be read twice. The swap RAM is on page 0x10 addresses 0x10 to 0x1f. The configuration software automatically writes this RAM. Currently there is no manual override within the configuration software.

The back end RAM encodes several fields into a 16 word by 16-bit RAM. The bits are mapped as shown in Table 18. The configuration software calculates the appropriate values for this RAM. There is no manual override option in the configuration software.

Table 18. Backward End Cell Control RAM Bit Map

BITS	DESCRIPTION
3..0	Backward end cell write address map for first iteration
7..4	Backward end cell write address map for second iteration
8	Backward end cell write enable
12..9	Backward end cell read address map
13	Blank feedback (end cell only) for odd symmetry first iteration
14	Blank feedback (end cell only) for odd symmetry second iteration
15	Unused

Programmable FIR, Gain, Transmit Input, and Receive Output Control Registers

The following tables detail the various control registers for a single PFIR filter. Note that the configuration software calculates most of these registers.

Table 19. Coefficient Address Generator Page 0x12 Address 0x10

Rcv	Tx	FIELD	BITS	Dflt	DESCRIPTION
E	E	*coef_ofsc*	3..0		FIR coefficient address offset
E	E	*coef_modc*	7..4		FIR coefficient modulo count
E	E	*coef_repc*	11..8		FIR coefficient repeat count
E	E	*coef_sym*	12		FIR hardware exploits symmetry (1) or not (0)
E	E	*coef_zerofr*	13		FIR zero forward read

Table 20. Common Address Generator Page 0x12 Address 0x11

Rcv	Tx	FIELD	BITS	Dflt	DESCRIPTION
E	E	*agen_recr*	3..0		FIR recirculate count
E	E	*agen_depd*	7..4		FIR depth count
E	E	*agen_modd*	11..8		FIR modulo count
E	E	*agen_togen*	12		FIR toggle enable for back end read and write

Table 21. Forward Read Address Generator Page 0x12 Address 0x12

Rcv	Tx	FIELD	BITS	Dflt	DESCRIPTION
E	E	*fragen_soff*	3..0		FIR forward read address generator offset
E	E	*fragen_srecr*	7..4		FIR forward read address generator recirculate count
E	E	*fragen_sdepd*	11..8		FIR forward read address generator depth count
E	E	*fragen_stepn*	15..12		FIR forward read address generator step

Table 22. Forward Write Address Generator Page 0x12 Address 0x13

Rcv	Tx	FIELD	BITS	Dflt	DESCRIPTION
E	E	*fwagen_soff*	3..0		FIR forward write address generator offset
E	E	*fwagen_srecr*	7..4		FIR forward write address generator recirculate count
E	E	*fwagen_sdepd*	11..8		FIR forward write address generator depth count
E	E	*fwagen_wrecrl*	15..12		FIR forward write address generator step

Table 23. Backward Read Address Generator Page 0x12 Address 0x14

Rcv	Tx	FIELD	BITS	Dflt	DESCRIPTION
E	E	*bragen_soff*	3..0		FIR backward read address generator offset
E	E	*bragen_srecr*	7..4		FIR backward read address generator recirculate count
E	E	*bragen_sdepd*	11..8		FIR backward read address generator depth count
E	E	*bragen_stepn*	15..12		FIR backward read address generator step

Table 24. Backward Write Address Generator Page 0x12 Address 0x15

Rcv	Tx	FIELD	BITS	Dflt	DESCRIPTION
E	E	*bwagen_soff*	3..0		FIR backward write address generator offset
E	E	*bwagen_srecr*	7..4		FIR backward write address generator recirculate count
E	E	*bwagen_sdepd*	11..8		FIR backward write address generator depth count
E	E	*bwagen_wrecrl*	15..12		FIR backward write address generator step

Table 25. Backward End Cell Read Address Generator Page 0x12 Address 0x16

Rcv	Tx	FIELD	BITS	Dflt	DESCRIPTION
E	E	*beragen_soff*	3..0		FIR backward end cell read address generator offset
E	E	*beragen_srecr*	7..4		FIR backward end cell read address generator recirculate count
E	E	*beragen_sdepd*	11..8		FIR backward end cell read address generator depth count
E	E	*beragen_stepn*	15..12		FIR backward end cell read address generator step

Table 26. Backward End Cell Write Address Generator Page 0x12 Address 0x17

Rcv	Tx	FIELD	BITS	Dflt	DESCRIPTION
E	E	*bewagen_soff*	3..0		FIR backward end cell write address generator offset
E	E	*bewagen_srecr*	7..4		FIR backward end cell write address generator recirculate count
E	E	*bewagen_sdepd*	11..8		FIR backward end cell write address generator depth count
E	E	*bewagen_wrecrl*	15..12		FIR backward end cell write address generator step

Table 27. Forward Write Strobe Page 0x12 Address 0x18

Rcv	Tx	FIELD	BITS	Dflt	DESCRIPTION
E	E	*fwa_strobe*	15..0		FIR forward write strobe

Table 28. Backward Write Strobe Page 0x12 Address 0x19

Rcv	Tx	FIELD	BITS	Dflt	DESCRIPTION
E	E	*bwa_strobe*	15..0		FIR backward write strobe

Table 29. Backward End Cell Read Bypass Page 0x12 Address 0x1a

Rcv	Tx	FIELD	BITS	Dflt	DESCRIPTION
E	E	*beagen_rbypass*	15..0		FIR backward end cell read bypass

Table 30. Transmit Input Formatter Page 0x13 Address 0x10

Rcv	Tx	FIELD	BITS	Dflt	DESCRIPTION
X	C	*tinf_bits*	2..0		Bits per word 0 − quiet, else bits = 4 x tinf_bits. Values 6 and 7 are illegal.
X	C	*tinf_pins*	5..3		Active pins quiet (0), four pins (1), 8 pins (2), 16 pins (4), illegal (else)
X	C	*tinf_iqmux*	6		IQ multiplexed (1) or real (0) inputs
X	C	*tinf_pariq*	7		IQ parallel (1) or not (0) I is bits 15..8 Q is bits 7..0
X	C	*tinf_src*	10..8		Source channel A normal (2),TDM (6), test (0); channels B−D normal (2), TDM (1), test (0)
C	C	*tinf_xmt*	11		Transmit (1) or receive (0)
C	C	*fso_sel*	13..12		Frame strobe output select quiet (0), xmt (1), receive (2), illegal (3)

Table 31. Transmit Frame Strobe Page 0x13 Address 0x11

Rcv	Tx	FIELD	BITS	Dflt	DESCRIPTION
X	E	*tinf_first*	3..0		Transmit input frame strobe load first word
X	E	*tinf_second*	7..4		Transmit input frame strobe load second word
D	D	*sck_div*	11..8	0	Slow clock divide minus 1
X	C	*tinf_fso_dly*	14..12		Transmit input frame strobe output delay values from 1 to 7

Table 32. Transmit Frame Counter Page 0x13 Address 0x12

Rcv	Tx	FIELD	BITS	Dflt	DESCRIPTION
X	E	*tinf_fs_cnt*	11..0		Transmit frame strobe count distance between frame strobes in divided clocks
X	E	*tinf_nfs*	15..12		Transmit number of frame strobes between *fir_swap* toggles

Table 33. *gain_lsb* Page 0x13 Address 0x13

Rcv	Tx	FIELD	BITS	Dflt	DESCRIPTION
C	C	*gain_lsb*	15..0		Lower 16 bits of gain (gain is scaled by 4096)

Table 34. Gain Controls Page 0x13 Address 0x14

Rcv	Tx	FIELD	BITS	Dflt	DESCRIPTION
C	C	*gain_msb*	2..0		Upper 3 bits of gain
D	D	*agc_freeze*	3	1	Freezes adaptive gain
C	C	*gain_rnd*	8..4		Rounds off bottom 0 to 16 bits of a 20-bit word
C	C	*gain_half*	9		Saturates gain output at 1/2. Used in xmt for symmetric filters
D	D	*agc_hold*	10	0	Gain tracks when low, holds when high for uP gain read back
D	D	*agc_gain_out*	11	0	Puts agc information into lower 8 bits of output data
D	D	*agc_zmag*	15..12	0	AGC mask for defining zero signal for faster AGC adaptation. Masks off bottom 0, 1, 2, 3, or 4 bits (0xf, e, c, 8, or 0)

Table 35. AGC Minimum Adaptation Limit Page 0x13 Address 0x15

Rcv	Tx	FIELD	BITS	Dflt	DESCRIPTION
D	D	*agc_min*	15..0	0	AGC minimum adaptation limit. Lower limit for adapted gain is *base_gain* − *agc_min*

Table 36. AGC Maximum Adaptation Limit Page 0x13 Address 0x16

Rcv	Tx	FIELD	BITS	Dflt	DESCRIPTION
D	D	*agc_max*	15..0	0	AGC maximum adaptation limit. Upper limit for adapted gain is *base_gain* + *agc_max*

Table 37. AGC Counts and Threshold Page 0x13 Address 0x17

Rcv	Tx	FIELD	BITS	Dflt	DESCRIPTION
D	D	*agc_zero_cnt*	3..0	0	Run of zeros before fast adaptation gain increase step Dzro is used
D	D	*agc_sat_cnt*	7..4	0	Run of saturated values before fast adaptation gain reduction step Dsat is used
D	D	*agc_thresh*	15..8	0	Threshold for AGC adaptation

Table 38. AGC Loop Gains Page 0x13 Address 0x18

Rcv	Tx	FIELD	BITS	Dflt	DESCRIPTION
D	D	*agc_Dblw*	3..0	0	Gain shift for below threshold
D	D	*agc_Dabv*	7..4	0	Gain shift for above threshold
D	D	*agc_Dzro*	11..8	0	Gain shift for zero data
D	D	*agc_Dsat*	15..12	0	Gain shift for saturated

Table 39. AGC Gain Read Back Page 0x13 Address 0x19

Rcv	Tx	FIELD	BITS	Dflt	DESCRIPTION
−	−	*gain_read*	15..0		Read back of current gain setting (read only)

Table 40. Power Meter (lsb) Page 0x13 Address 0x1a

Rcv	Tx	FIELD	BITS	Dflt	DESCRIPTION
−	−	*pwr_meter_lsb*	15..0		Read back of power meter least significant 16 bits (read only)

Table 41. Power Meter (msb) Page 0x13 Address 0x1b

Rcv	Tx	FIELD	BITS	Dflt	DESCRIPTION
−	−	*pwr_meter_msb*	15..0		Read back of power meter most significant 16 bits (read only)

Table 42. Power Meter Status Page 0x13 Address 0x1c

Rcv	Tx	FIELD	BITS	Dflt	DESCRIPTION
−	−	*ready*	15		1 means power sample ready to read, user must reset to 0
−	−	*missed*	14		Set if power sample missed; reset by user
−	−	*too_soon*	13		Set if power sample reads too soon; reset by user
			12		Factory test bit keep 0
−	−	*pwr_os*	11		One shot control keep 0 if *msr* = 1, if *msr* = 0, then use to fire one shot. 0 = armed, 1 = fire; be sure that there are 2 clocks between arming and firing
D	D	*pwr_msr*	10	0	1 = use read of ms16 power (address 0x1b) to fire one shot; 0 = manually fire one shot
			9..8		Unused
−	−	*gain_sign*	7		Read only, sign bit from gain. Only useful if one collects statistics over time.
			6		Sticky bit, set if gain zro = 1, reset by user
			5		Sticky bit, set if gain sat = 1, reset by user
			4..0		Unused

Table 43. Power Meter Integration Time Page 0x13 Address 0x1d

Rcv	Tx	FIELD	BITS	Dflt	DESCRIPTION
D	D	*pwr_mtr_integ*	15..0	0	Power meter integration time in words

Table 44. Receive Output Formatter Page 0x13 Address 0x1e

Rcv	Tx	FIELD	BITS	Dflt	DESCRIPTION
C	X	*routf_pins*	2..0		Receive output formatter active pins. Active pins are always the msbs. Set to 1, 2, or 4 for 4, 8, or 16 active pins.
C	X	*routf_bits*	7..3		Receive output formatter bits in a word. Set to 1, 2, 4, 8, 16, or 32 to get a word size of 4, 8, 12, 16, or 20.
C	X	*routf_iqmux*	8		Receive output complex (1) or real (0) per port. Note that when *splitiq* is active, *routf_iqmux* should be set to real.
C	C	*routf_pwrdown*	9		Power down receive output formatter block.
C	X	*routf_ctdm*	11		Activate TDM receive output.
D	D	*pwr_test*	12	0	Test mode for power meter. 0 for normal operation
D	D	*tx_pat_gen*	14..13	0	Transmit pattern generator enable and source selection off (0), random (1), constant 0x4000 (2), random with bit 14 inverted (3).

Table 45. Receive Checksum Page 0x13 Address 0x1f

Rcv	Tx	FIELD	BITS	Dflt	DESCRIPTION
−	−	*Rcv_checksum*	15..0	−	Checksum results for receive (read only).

Table 46. FIR Swap Ram Controls Page 0x14 Address 0x10

Rcv	Tx	FIELD	BITS	Dflt	DESCRIPTION
E	E	*swap_wa_cnt*	3..0		FIR swap write address down count
E	E	*swap_ra_cnt*	7..4		FIR swap read address down count
E	E	*swap_xmt*	8		FIR swap RAM in transmit (1) or receive (0)
E	E	*swap_cmplx*	9		FIR swap expects complex from CIC (1) or real (0)
E	E	*fir_fb*	10		FIR forward broadcast (1) or not (0)
E	E	*swap_fb*	11		FIR swap RAM in forward broadcast (1) or not (0)
E	E	*swap_rcv_tdly*	12		FIR swap RAM. Reduces cic to FIR data valid delay by 1 when cic is bypassed – helps with data alignment in certain cases. Calculated by software.

Table 47. FIR Accumulator Controls Page 0x14 Address 0x11

Rcv	Tx	FIELD	BITS	Dflt	DESCRIPTION
E	E	*acc_dly*	3..0		FIR accumulator delay minus 1
E	E	*acc_cnt*	7..4		FIR number of partial products to accumulate
E	E	*acc_dly0*	8		FIR no accumulator delay (1) or not (0)
E	E	*acc_bypass*	9		FIR accumulator bypass (1) or normal (0). Must also set *acc_cnt* to zero for bypass.
E	E	*acc_enram*	10		FIR enable accumulate ram (1) normal. Set to 0 when *acc_dly0* is 1.

Table 48. FIR Output Page 0x14 Address 0x12

Rcv	Tx	FIELD	BITS	Dflt	DESCRIPTION
C	C	*fir_shift*	2..0		FIR shift up. Increase filter gain
C	C	*fir_xenq*	3		FIR transmit enable q. Set for transmit and complex filtering.
C	C	*fir_xeni*	4		FIR transmit enable i. Set for transmit, clear in receive.
C	C	*fir_half*	5		FIR half scale. Set to 0.
C	C	*fir_rnd20*	6		FIR round For receive, set to 1 for 20 bits to gain. In transmit, set to 0 for 18 bits to CIC.
C	C	*fir_cmplx*	7		FIR outputs complex data. Normally set, clear for *splitiq* or some bypass modes.

Table 49. FIR Sync Count Page 0x14 Address 0x13

Rcv	Tx	FIELD	BITS	Dflt	DESCRIPTION
E	E	*fir_sync_cnt*	15..0		FIR internal sync count to periodically resync various FIR counters

Table 50. FIR Clock Page 0x14 Address 0x14

Rcv	Tx	FIELD	Bits	Dflt	Description
E	E	*fir_clk_cnt*	7..0		Number of FIR clocks per CIC input (xmt) or output (rcv) minus 1
E	E	*fir_clk_cnt_en*	8		1 for normal operation; set to 0 and *fir_clk_cnt*+1 for FIR clock always on
E	E	*fir_test*	9		Set to 0 for normal operation
E	E	*fir_en_sync_cnt*	10		Enable fir sync counter (1)
D	D	*cksum_sync_back*	14..12	7	Receive checksum sync and transmit pattern generator sync

Table 51. Channel Syncs Page 0x14 Address 0x15

Rcv	Tx	FIELD	BITS	Dflt	DESCRIPTION
D	D	*fir_sync*	2..0	2	Sync source selection for FIR
D	D	*coef_sync*	5..3	6	Sync source selection for coefficient swapping. Used with multiple coefficient sets.
D	D	*gain_sync*	8..6	6	Sync source selection for gain updates
D	D	*pwr_mtr_sync*	11..9	7	Sync source selection for starting power meter
D	D	*sck_sync*	14..12	4	Sync source selection for slow clock divider

CIC and MIX Control Registers

There are two cicmix blocks. Controls on page 0x80 addresses 0x10−0x17 control channel A, while addresses 0x18 to 0x1f control channel B. The controls for channels C and D are located on page 0xa0. The bit fields correspond so the detailed descriptions are not repeated.

Table 52. CIC Mode Page 0x80 Address 0x10

Rcv	Tx	FIELD	BITS	Dflt	DESCRIPTION
C	C	*cic_shift*	5..0		CIC is followed by a shifter by a shifter to compensate for CIC gain. 0 ≤ *cic_shift* ≤ 39.
C	C	*cic_rcv*	7..6		CIC mode is quiet (0), receive (1), transmit (2), or illegal (3)
X	C	*cic_xmt_5stg*	8		CIC transmits five stages (1), rather than the normal six stages (0)
C	X	*cic_rshift*	9		Upshift 1 bit in receive mode after CIC filtering before rounding
C	C	*cic_2x*	10		Operate the CIC in double rate mode
X	C	*cic_xmt_d6stg*	11		Must be set for CIC in double rate and six stage, else clear
C	X	*cic_rcv_cross*	12		Use cross-strapped CIC inputs in receive. Set for *splitiq* mode
C	X	*cic_rcv_full*	13		Saturate CIC output at full scale (1) for nonsymmetrical FIR or half scale (0) for symmetric FIR. Only affects receive outputs.
C	C	*cic_bypass*	14		Bypass CIC in transmit (1), must also set *cic_shift* (39) and *ncic* (0). Clear in receive.
−	−	*cic_fl_status*	15		CIC flush status sticky bit. Chip sets if autoflushed. User must clear.

Table 53. Phase Page 0x80 Address 0x11

Rcv	Tx	FIELD	BITS	Dflt	DESCRIPTION
D	D	*phase*	15..0	0	Phase is phase/2^{16} Hz

Table 54. Frequency (lsb) Page 0x80 Address 0x12

Rcv	Tx	FIELD	BITS	Dflt	DESCRIPTION
D	D	*freq_lsb*	15..0	0	Bottom 16 bits of frequency

Table 55. Frequency (mid) Page 0x80 Address 0x13

Rcv	Tx	FIELD	BITS	Dflt	DESCRIPTION
D	D	*freq_mid*	15..0	0	Middle 16 bits of frequency

Table 56. Frequency (msb) Page 0x80 Address 0x14

Rcv	Tx	FIELD	BITS	Dflt	DESCRIPTION
M	M	*freq_msb*	15..0		Top 16 bits of frequency. Values in Table 54 through Table 56 may be set using the pseudo field *freq*.

Table 57. Mixer Page 0x80 Address 0x15

Rcv	Tx	FIELD	BITS	Dflt	DESCRIPTION
M	X	*mix_rcv_sel*	1..0		Mixer input selection in receive (paths a, b, c, or d for 0, 1, 2, or 3 respectively).
C	X	*mix_rcv_cmplx*	2		Set for receive complex input at full rate or double rate.
C	C	*mix_inv_qsin*	4		Invert the output of Qdata by sin in the mixer.
C	C	*mix_qsin*	6..5		Select data input to qsin multiplier as zero (0), receive (1), transmit cross-strapped (2), or transmit (3). Transmit cross-strapped is for transmit in *splitiq* or double rate modes.
C	C	*mix_inv_qcos*	7		Invert the output of Qdata by cos in the mixer.
C	C	*mix_qcos*	9..8		Select data input to qsin multiplier as zero (0), receive (1), transmit cross-strapped (2), or transmit (3). Transmit cross-strapped is for transmit in *splitiq* or double rate modes.
C	C	*mix_inv_isin*	10		Invert the output of Idata by sin in the mixer.
C	C	*mix_isin*	12..11		Select data input to qsin multiplier as zero (0), receive (1), transmit cross-strapped (2), or transmit (3). Transmit cross-strapped is for transmit in *splitiq* or double rate modes.
C	C	*mix_inv_icos*	13		Invert the output of Idata by cos in the mixer.
C	C	*mix_icos*	15..14		Select data input to qsin multiplier as zero (0), receive (1), transmit cross-strapped (2), or transmit (3). Transmit cross-strapped is for transmit in *splitiq* or double rate modes.

Table 58. NCO Syncs Page 0x80 Address 0x16

Rcv	Tx	FIELD	BITS	Dflt	DESCRIPTION
D	D	*freq_sync*	2..0	6	Sync source for frequency word update
D	D	*phase_sync*	5..3	6	Sync source for phase word update
D	D	*dith_sync*	8..6	0	Sync source for dither update. Typically set to never. Set to always to disable.
D	D	*nco_sync*	11..9	0	Sync source for nco word update. Disrupts signal processing when sync occurs
D	D	*flush_sync*	14..12	0	Sync source for cic flush. Disrupts signal processing when sync occurs.
D	D	*dith_test*	15	0	Experimental dither mode, set to 0.

Table 59. CIC Count and Sync Page 0x80 Address 0x17

Rcv	Tx	FIELD	BITS	Dflt	DESCRIPTION
C	C	*ncic*	11..0		CIC decimation/interp count minus one
D	D	*cic_sync*	14..12	2	Sync source for cic counter. Should be periodic in (*ncic* + 1)

Registers on page 0x81 are for transmit output summing and receiver desensitizing.

Table 60. Sum Tree Sum Selection Page 0x81 Address 0x10

Rcv	Tx	FIELD	BITS	Dflt	DESCRIPTION
X	C	*sum_selI*	7..0		Selects inputs for I sum both A and B paths.
X	C	*sum_selQ*	15..8		Selects inputs for Q sum both A and B paths.

Table 61. Sum Tree Multiplexing Page 0x81 Address 0x11

Rcv	Tx	FIELD	BITS	Dflt	DESCRIPTION
X	C	*sum_shift*	2..0		Upshifts sum before output (0−7).
X	C	*sum_ia*	4..3		Summing mode off (0), 22 bit sumin (1), bypass (2), 16 bit sumin (3)
X	C	*sum_ib*	6..5		Summing mode off (0), 22 bit sumin (1), bypass (2), 16 bit sumin (3)
X	C	*sum_qa*	8..7		Summing mode off (0), 22 bit sumin (1), bypass (2), 16 bit sumin (3)
X	C	*sum_qb*	9		Summing mode normal (0), quiet (1)
X	C	*sum_in*	11..10		Sum in mode off (0), IQ multiplexed (1), 16 bit (2), 22 bit (3)
−	−	*sum_of_Ia*	12		Sum tree overflow sticky status bit for Ia
−	−	*sum_of_Ib*	13		Sum tree overflow sticky status bit for Ib
−	−	*sum_of_Qa*	14		Sum tree overflow sticky status bit for Qa
−	−	*sum_of_Qb*	15		Sum tree overflow sticky status bit for Qb

Table 62. Receive Sensitivity Reduction Path A Page 0x81 Address 0x12

Rcv	Tx	FIELD	BITS	Dflt	DESCRIPTION
D	X	*rcv_noise_A*	15..0	0	Adds noise to input A to desensitize the digital down converters. Must also release pn generator sync and set test mode to random.

Table 63. Receive Sensitivity Reduction Path B Page 0x81 Address 0x13

Rcv	Tx	FIELD	BITS	Dflt	DESCRIPTION
D	X	*rcv_noise_B*	15..0	0	Adds noise to input B to desensitize the digital down converters. Must also release pn generator sync and set test mode to random.

Table 64. Receive Sensitivity Reduction Path C Page 0x81 Address 0x14

Rcv	Tx	FIELD	BITS	Dflt	DESCRIPTION
D	X	*rcv_noise_C*	15..0	0	Adds noise to input C to desensitize the digital down converters. Must also release pn generator sync and set test mode to random.

Table 65. Receive Sensitivity Reduction Path D Page 0x81 Address 0x15

Rcv	Tx	FIELD	BITS	Dflt	DESCRIPTION
D	X	*rcv_noise_D*	15..0	0	Adds noise to input D to desensitize the digital down converters. Must also release pn generator sync and set test mode to random.

Registers on page 0xa1 control receive input formatting and transmit output formatting.

Table 66. Receive Input Formatter Page 0xa1 Address 0x10

Rcv	Tx	FIELD	BITS	Dflt	DESCRIPTION
C	X	*rinf_sel_A*	3..0		Receive input formatting. Normal usage quiet (0), Q part of nonmultiplexed complex (1), IQ multiplexed (3), real or real portion of nonmultiplexed complex (4), and test (8). Detailed settings are quiet (0), input to Q (1), input delayed by 1 to I (2), input to I (4), and test to both I and Q (8).
C	X	*rinf_sel_B*	7..4		Same as above, but for path B
C	X	*rinf_sel_C*	11..8		Same as above, but for path C
C	X	*rinf_sel_D*	15..12		Same as above, but for path D

Table 67. General Timer Page 0xa1 Address 0x11

Rcv	Tx	FIELD	BITS	Dflt	DESCRIPTION
D	D	*gen_timer*	15..0	65535	General-purpose timer. A 24-bit counter the bottom eight bits of the count value are normally assumed to be zero. Timer counts down from 256 x (*gen_timer* + 1) to zero, then repeats. The timer starts over with every sync, provides ramp data for receive testing, and outputs a pulse at restart plus five clocks.

Table 68. Receive Syncs Page 0xa1 Address 0x12

Rcv	Tx	FIELD	BITS	Dflt	DESCRIPTION
D	X	*rinf_zpad*	3..0	0	Receive input formatter zero pad count. Effectively interpolates by (*rinf_zpad*+1) up to the CK rate.
D	X	*rinf_diag*	5..4	0	Receive diagnostic test pattern ramp (0), 0 (1), random (2), constant 0x4000 (3).
D	D	*gen_timer_test*	6	0	Shorten *gen_timer* for test (1) (bottom 8 bits are forced high). Normally 0.
D	D	*gen_timer_sync*	9..7	7	Sync control for general timer
D	D	*cksum_sync_front*	12..10	7	Sync control for receive input pattern generator and transmit check sum.
D	X	*rinf_zpad_sync*	15..13	7	Sync control for receive zero padding

Table 69. Transmit Output Multiplexing Page 0xa1 Address 0x13

Rcv	Tx	FIELD	Bits	Dflt	Description
X	D	*toutf_hold_sync*	2..0	7	Sync control for transmit output hold
X	C	*toutf_do*	8..4		Transmit output format control for DO. quiet (0), id (1), qb (2), ib lsb's (4), qa lsb's (8), CO complement (16). Others illegal.
X	C	*toutf_co*	11..9		Transmit output format control for CO. quiet (0), ic (1), ib (2), qa (4).
X	C	*toutf_bo*	15..12		Transmit output format control for BO. quiet (0), ib (1), qa (2), lsb of ia (4), or AO complement (8).

Table 70. Transmit Output Round and Hold Page 0xa1 Address 0x14

Rcv	Tx	FIELD	BITS	Dflt	DESCRIPTION
X	C	*toutf_rnd_AB*	1..0		Round transmit outputs AO, and BO to 16 (1), 14 (2), 12 (3) bits, or no round (0).
X	D	*toutf_offsetbin*	3	0	Offset binary format (1) or two's complement (0). Common to all transmit outputs.
X	C	*toutf_halfcmplx_AB*	4		Time multiplex IQ data by decimating by two at output.
X	D	*toutf_hold*	7..5	0	Hold output for toutf_hold+1 clocks − effectively decimating the signal.
X	C	*toutf_rnd_CD*	9..8		Round transmit outputs for CO and DO to 16 (1), 14 (2), 12 (3), bits or no round (0).
X	C	*toutf_quiet_CD*	10		Quiet CD path (1) when unused.
X	C	*toutf_halfcmplx_CD*	11		Time multiplex IQ for channels C and D.
E	C	*toutf_xmt_CD*	12		Channels C and D setup in transmit (1) or receive (0).
X	C	*toutf_sumin*	13		CO and DO setup as inputs for sumin (1), or outputs (0).
E	C	*toutf_xmt_AB*	14		Channels A and B setup in transmit (1) or receive (0).

Table 71. Transmit Checksum Page 0xa1 Address 0x15

Rcv	Tx	FIELD	BITS	Dflt	DESCRIPTION
−	−	*transmit_cksum*	15..0		Checksum results for transmit (read only)

EXAMPLES

CDMA2000

This section describes an example of the down-conversion filter response for CDMA2000 1X. The GC5016 configuration values were input sample rate of 78.643 MSPS, CIC decimation of eight, and PFIR decimation of four for an overall decimation of 32, output rate of 2.4576 MSPS (2x chip rate), and 255 PFIR taps. The overall filter response, including both the CIC and PFIR filters, is shown in Figure 31 and Figure 32. As seen in Figure 32 of the transition region, the filter response meets the CDMA2000 1X stop-band rejection requirements of –50 dB at 750 kHz and –87 dB at 900 kHz.

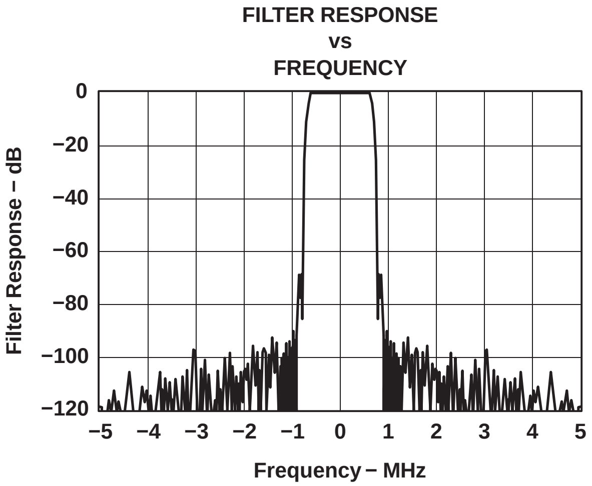


Figure 31. CDMA2000 1X Filter Response

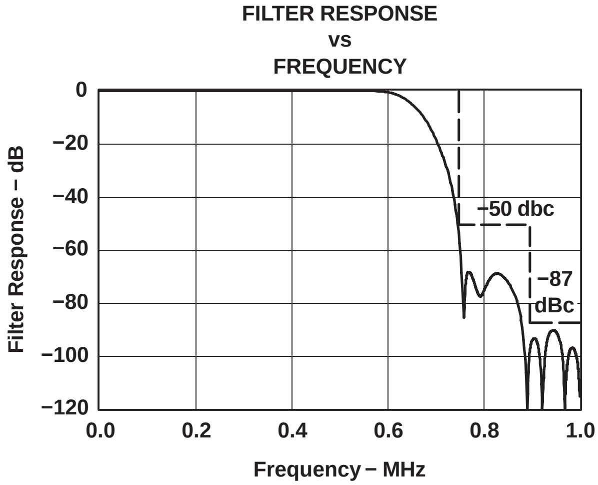


Figure 32. CDMA2000 1X Filter Response Transition Region with Spectral Mask

In transmit the input is presumed to operate at one sample per chip (1.2288 MHz), the PFIR interpolates by four and uses a 192 tap filter that applies the phase predistortion, pulse shaping, adjacent channel rejection, and CIC roll-off compensation. This is followed by a six stage CIC filter. The cmd5016 configuration file and filter taps are available on the web.

WCDMA (UMTS)

This section describes an example of the down-conversion filter response for WCDMA. The GC5016 configuration values were input sample rate of 122.88 MHz, a CIC and PFIR decimation of four each for an overall decimation of 16, output rate of 7.68 MSPS, and 255 PFIR taps. The overall filter response filter, including both the CIC and PFIR filters, an optimized raised root cosine filter with α = 0.22, is shown in Figure 33 and Figure 34. The stop-band attenuation is better than –80 dBc for frequencies more than 2.5 MHz from the band center.

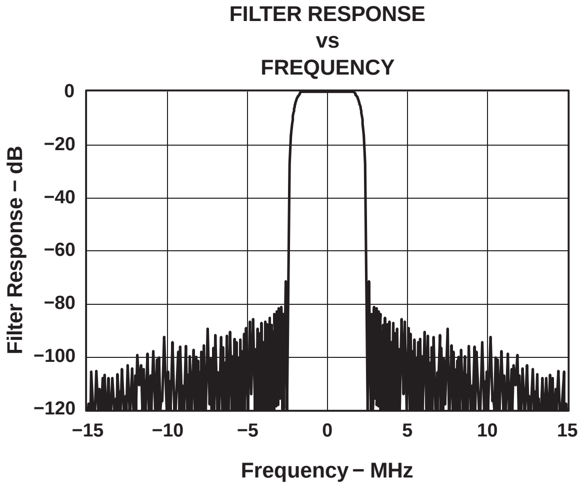


Figure 33. UMTS Filter Response

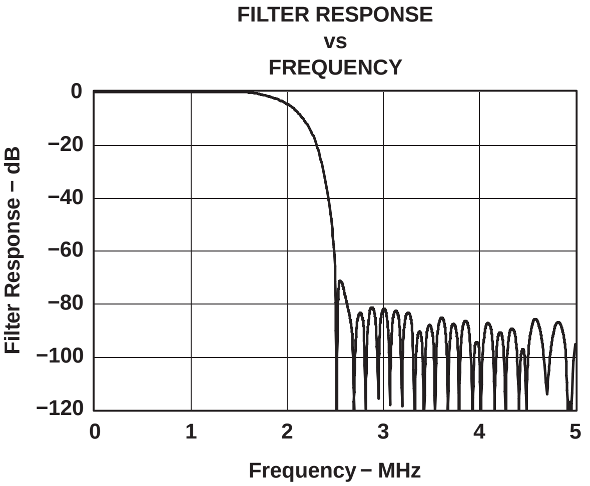


Figure 34. UMTS Filter Response Transition Region

The cmd5016 configuration file and filter taps are available on the web.

APPLICATION INFORMATION

BOARD BRING-UP PROCEDURE

This section describes a recommended procedure for checkout of a board using the GC5016. The various test files are available on the website.

Basic Control Path

Write reset value (0xffff) to register 0. Read back to see 0xfff[c−f]. The bottom two bits are status bits and can be any value. Write and read the page register (address 0x2). The lower byte should be just what you write in. The upper byte should read back the revision (currently 1), regardless of what was written. If possible, use a scope to capture the event so you can confirm setup/hold, output delay, strobe pulse width, voltage levels, and signal integrity.

Thorough Control Path Test

Use the *control_check.gc101* script to read and write every control register and coefficient RAM in the chip with all 0's, 1's, 5's and A's and it then tests to see that the proper results are returned. Two commands are used (dwr16 and dcm16) as shown in the following table:

dwr16 address data	Write to the chip (both address and data are in hex)
dcm16 address mask *expected_data*	Reads from the chip, masks the results, and checks against the expected data.

The software should execute code similar to the following:

if(((*read_results*^*expected_data*) and mask) != 0) complain

At this point the user should be confident in the control interface.

Built-in Self-test

These built-in self-tests provide the chip with input data using an internal pattern generator, an internal sync using the general timer in the chip, and analyzes the output using a checksum generator. These tests depend on the board to provide a solid control path, good clock, and good power. They generally work by setting the chip into a particular mode, then running the patterns (typically for at least four million clocks), then reading out the checksum result. Four checksum configurations are provided to provide good coverage of chip internals. If possible, use a scope to check the quality of the clock, power, and ground. Table 72 provides the addresses to find the resultant checksums.

Table 72. Checksum Addresses

	Transmit	Receive Channel A	Receive Channel B	Receive Channel C	Receive Channel D
Page	0xa1	0x13	0x33	0x53	0x73
Address	0x15	0x1f	0x1f	0x1f	0x1f

The user should configure the chip as specified in the configuration file, wait the recommended time, and then read the checksum results and compare them to the expected results shown in Table 73.

Table 73. Expected Checksum Results

Configuration Name	Wait Time (in clocks)	Expected Results				
		Transmit	Chan A	Chan B	Chan C	Chan D
TCK10R0.GC101	4,000,000	0x121D	−	−	−	−
TCK100R0.GC101	4,000,000	0x61EF	−	−	−	−
RCK100R0.GC101	4,000,000	−	0x64CF	0x4747	0x7BD9	0x4747
TRCK100R0.GC101	4,000,000	0x15C8	0x1C1B	0x030D	0x7BD9	0xE1CB

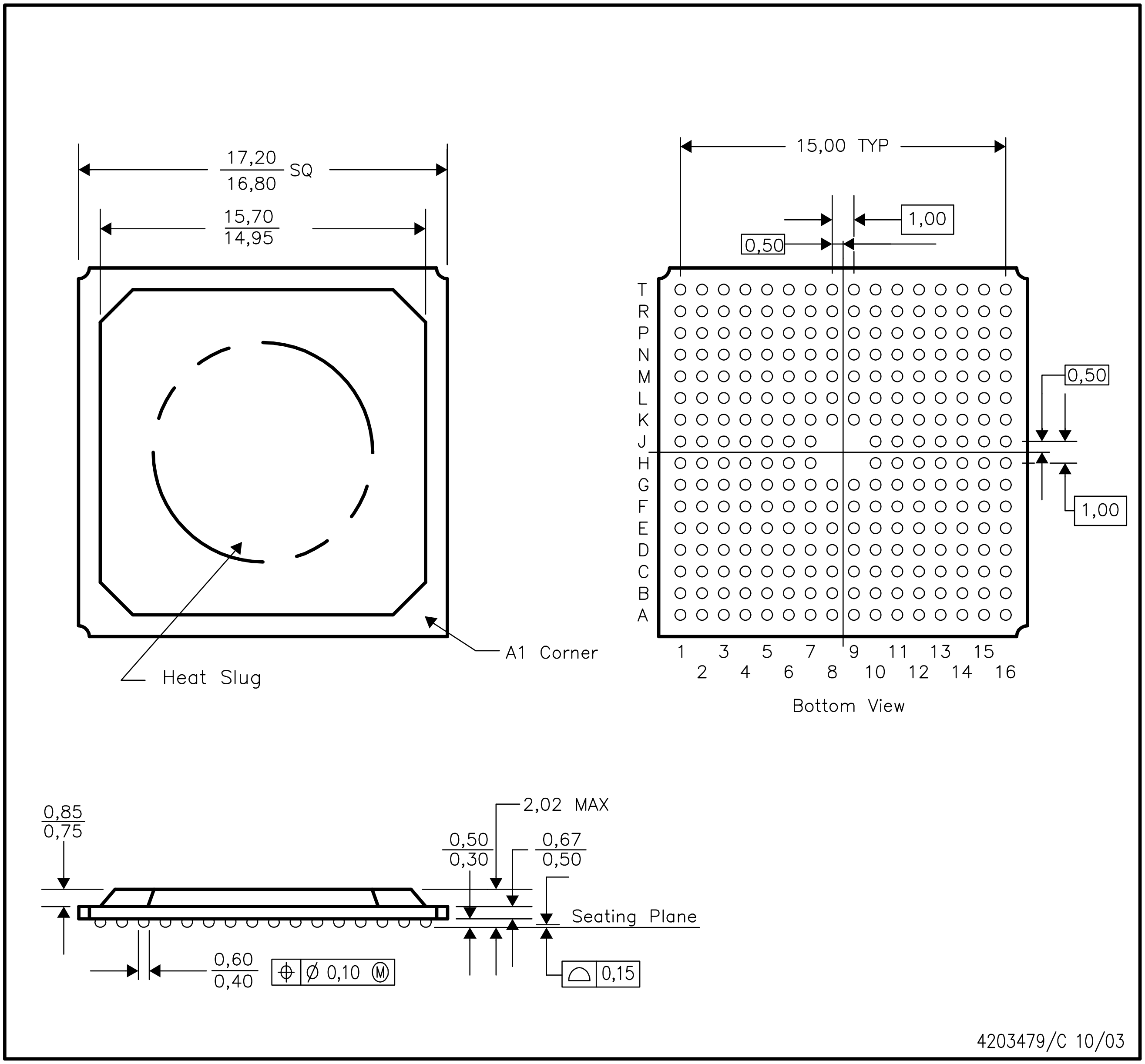
Output Test Configuration

In these tests the chip is setup using the internal pattern generator to provide a simple sin wave output regardless of the input. There is a transmit configuration (*tsin_r0*) and a receive configuration (*rsin_r0*). The transmit configuration outputs a single real tone on each of the four output ports (AO, BO, CO, and DO) at a rate of one new sample for each clock. The frequencies are 0.0633, 0.0711, 0.1297, and 0.1378 respectively. The receive configuration outputs one complex tone on each port as I followed by Q followed by 14 zeros. The four frequencies are 0.0249, 0.0498, 0.0996, and 0.1992 (of Fsout). In both cases, $\overline{\text{SO}}$ can be used to synchronize the capture of the output if desired – its period is 2^20 clocks. The frequencies have been chosen so that the output is also periodic in 2^20 to avoid glitches at the repetition point. While the output is periodic in a general sense, it is not digitally precise periodic due to dithering of the NCO. Use a scope to check that output data from GC5016 provides sufficient setup and hold and signal integrity for the receiving device.

Input Test Configuration

In this configuration (*rby_r0*), the chip output is the same as the chip input, only delayed by 50 cycles. This test can be used to be sure that the input is read properly by the GC5016. Use a scope to check that input setup and hold times are met and that the signal doesn’t have excessive ringing.

GDJ (S-PBGA-N252) PLASTIC BALL GRID ARRAY



NOTES:
- A. All linear dimensions are in millimeters.
- B. This drawing is subject to change without notice.
- C. Thermally enhanced plastic package with heat slug (HSL).

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