

ARTISAN®
TECHNOLOGY GROUP

Your **definitive** source for quality pre-owned equipment.

Artisan Technology Group
(217) 352-9330 | sales@artisantg.com | artisantg.com

Full-service, independent repair center
with experienced engineers and technicians on staff.

We buy your excess, underutilized, and idle equipment
along with credit for buybacks and trade-ins.

Custom engineering
so your equipment works exactly as you specify.

- Critical and expedited services
- In stock / Ready-to-ship
- Leasing / Rentals / Demos
- ITAR-certified secure asset solutions

Expert team | Trust guarantee | 100% satisfaction

All trademarks, brand names, and brands appearing herein are the property of their respective owners.

Find the *Pentek 6509* at our website: *Click HERE*

Appendix A: Programming Examples

A.1 6509.h

```
/*************************************************************************
*
* File : 6509.h
*
* Description: This file contains the 6509 Memory Map, Register
* Definitions, and function prototypes.
*
* $Revision:$ $Date:$
*
* $History:$
*
*************************************************************************/

#ifndef P6509_H
#define P6509_H

/* Board Configuration Parameter Constants */

#define P6509_NUM_RCVRS 8
#define P6509_NUM_COMM_PORTS 4
#define P6509_NUM_INPUT_CHANNELS 2
#define P6509_FIFO_SIZE 1024

/* 6509 Local C40 Reg Base - This is the Local C40 Address for accessing
                              the 6509's SRAM and Registers                */

#define P6509_C40_REG_BASE 0x80000000

/* 6509 Memory Map. This addresses are byte offsets from the VMEBase. To use
these offsets from the local C40, they must be divided by 4.                 */

#define P6509_GLOBAL_SRAM 0x00000000
#define P6509_INTR_MASK_REG1 0x00080000
#define P6509_INTR_MASK_REG2 0x00088000
#define P6509_STATUS_REG1 0x00090000
#define P6509_STATUS_REG2 0x00098000
#define P6509_FIFO_RESET_REG 0x000A0000
#define P6509_COMM_PORT_CNTL_REG 0x000A8000
#define P6509_GRAY_1_BASE 0x000B8000
#define P6509_GRAY_2_BASE 0x000B8040
#define P6509_GRAY_3_BASE 0x000B8080
#define P6509_GRAY_4_BASE 0x000B80C0
#define P6509_GRAY_5_BASE 0x000B8100
#define P6509_GRAY_6_BASE 0x000B8140
#define P6509_GRAY_7_BASE 0x000B8180
#define P6509_GRAY_8_BASE 0x000B81C0
#define P6509_GRAY_ALL_BASE 0x000B8200
#define P6509_MODE_SLCT_REG 0x000B8240
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

A.1 **6509.h** (continued)

```
#define P6509_TAG_ENABLE_REG 0x000B8280
#define P6509_INPUT_MUX_REG 0x000B8300
#define P6509_THRSHLD_DETECT_A 0x000B8340
#define P6509_THRSHLD_DETECT_B 0x000B8380
#define P6509_RCVR_1_FIFO 0x000C0000
#define P6509_RCVR_2_FIFO 0x000C8000
#define P6509_RCVR_3_FIFO 0x000D0000
#define P6509_RCVR_4_FIFO 0x000D8000
#define P6509_RCVR_5_FIFO 0x000E0000
#define P6509_RCVR_6_FIFO 0x000E8000
#define P6509_RCVR_7_FIFO 0x000F0000
#define P6509_RCVR_8_FIFO 0x000F8000
#define P6509_INTR_ACKN_REG 0x00100000
#define P6509_VME_INTR_REG 0x00140000
#define P6509_GLOBAL_FLASH 0x00200000

/* Constants specific to each 6509 Registers are listed below */

/* Interrupt Mask Register 2 */

#define P6509_IM2_FIFO_NOT_EMPTY_1 0x00000001
#define P6509_IM2_FIFO_NOT_EMPTY_2 0x00000002
#define P6509_IM2_FIFO_NOT_EMPTY_3 0x00000004
#define P6509_IM2_FIFO_NOT_EMPTY_4 0x00000008
#define P6509_IM2_FIFO_NOT_EMPTY_5 0x00000010
#define P6509_IM2_FIFO_NOT_EMPTY_6 0x00000020
#define P6509_IM2_FIFO_NOT_EMPTY_7 0x00000040
#define P6509_IM2_FIFO_NOT_EMPTY_8 0x00000080
#define P6509_IM2_FIFO_HALF_FULL_1 0x00000100
#define P6509_IM2_FIFO_HALF_FULL_2 0x00000200
#define P6509_IM2_FIFO_HALF_FULL_3 0x00000400
#define P6509_IM2_FIFO_HALF_FULL_4 0x00000800
#define P6509_IM2_FIFO_HALF_FULL_5 0x00001000
#define P6509_IM2_FIFO_HALF_FULL_6 0x00002000
#define P6509_IM2_FIFO_HALF_FULL_7 0x00004000
#define P6509_IM2_FIFO_HALF_FULL_8 0x00008000

/* Interrupt Mask Register 1 */

#define P6509_IM1_FIFO_FULL_1 0x00000001
#define P6509_IM1_FIFO_FULL_2 0x00000002
#define P6509_IM1_FIFO_FULL_3 0x00000004
#define P6509_IM1_FIFO_FULL_4 0x00000008
#define P6509_IM1_FIFO_FULL_5 0x00000010
#define P6509_IM1_FIFO_FULL_6 0x00000020
#define P6509_IM1_FIFO_FULL_7 0x00000040
#define P6509_IM1_FIFO_FULL_8 0x00000080
#define P6509_IM1_CLOCK_NOT_READY_A 0x00000100
#define P6509_IM1_CLOCK_NOT_READY_B 0x00001000
#define P6509_IM1_THRSHLD_DETECT_A 0x00010000
#define P6509_IM1_THRSHLD_DETECT_B 0x00100000
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

A.1 6509.h (continued)

```
/* Definitions of Gray Chip Channel Registers - Each channel has 16 Registers */
/* The Constant are offsets from P6509_GRAY_1_BASE */

#define P6509_CH_FREQ_BYTE_0 0x00000000L /* Lsb */
#define P6509_CH_FREQ_BYTE_1 0x00000001L
#define P6509_CH_FREQ_BYTE_2 0x00000002L
#define P6509_CH_FREQ_BYTE_3 0x00000003L /* Msb */
#define P6509_CH_SYNC_MODE 0x00000004L
#define P6509_CH_OUTPUT_MODE 0x00000005L
#define P6509_CH_FILTER_MODE 0x00000006L
#define P6509_CH_GAIN_CONTROL 0x00000007L
#define P6509_CH_DECIMATION_LSB 0x00000008L
#define P6509_CH_DECIMATION_MSB 0x00000009L
#define P6509_CH_OUTPUT_STATUS 0x0000000aL
#define P6509_CH_CHECKSUM 0x0000000bL
#define P6509_CH_I_OUTPUT_BYTE_0 0x0000000cL
#define P6509_CH_I_OUTPUT_BYTE_1 0x0000000dL
#define P6509_CH_Q_OUTPUT_BYTE_0 0x0000000eL
#define P6509_CH_Q_OUTPUT_BYTE_1 0x0000000fL

/* P6509_RCVR_PARAMS - This parameter structure contains all the
                       settings specific to setting up a receiver.         */

typedef struct P6509_RCVR_PARAMS
{
unsigned long decimation_rate;
unsigned long data_mode;
unsigned long data_source;
unsigned long packing_mode;
unsigned long data_order;
unsigned long data_transfer;
unsigned long rcvr_select;
unsigned long tag_enable;
unsigned long flip_spectrum;
unsigned long offset_spectrum;
unsigned long filter_select;
unsigned long rcvr_sync_mode;
float lo;
float input_clock_rate;
unsigned long spare[5];
} P6509_RCVR_PARAMS;

/* P6509_INPUT_PARAMS - This structure defines parameters specific to
                        Input Channels.                                    */

typedef struct P6509_INPUT_PARAMS
{
unsigned long threshold_level;
unsigned long threshold_cross_cnt;
unsigned long input_sync_mode;
} P6509_INPUT_PARAMS;
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

A.1 6509.h (continued)

```
/* P6509_REG_ADDR - This structure contains the 6509's register addresses. */

typedef struct P6509_REG_ADDR
{
volatile unsigned long *Int_Mask_1;
volatile unsigned long *Int_Mask_2;
volatile unsigned long *Stat_Reg_1;
volatile unsigned long *Stat_Reg_2;
volatile unsigned long *Fifo_Reset;
volatile unsigned long *Out_Comm_Ctl;
volatile unsigned long *Gray_1_Base;
volatile unsigned long *Mode_Slct;
volatile unsigned long *Tag_En;
volatile unsigned long *In_Mux;
volatile unsigned long *Thrshld_Detect_A;
volatile unsigned long *Thrshld_Detect_B;
volatile unsigned long *Fifo;
volatile unsigned long *Intr_Ackn_Addr;
volatile unsigned long *VME_Intr_Reg;
} P6509_REG_ADDR;

/* 6509 Function Prototypes */

void P6509InitRegAddr(unsigned long regBaseAddr, P6509_REG_ADDR *p6509Reg);
void P6509InitRcvrRegs(P6509_RCVR_PARAMS *rcvr_params,
                       P6509_REG_ADDR *p6509Reg, unsigned int rcvr_mask);
void P6509InitInputRegs(P6509_INPUT_PARAMS *input_params,
                        P6509_REG_ADDR *p6509Reg, unsigned int input_mask);
void P6509ResetRegs(P6509_REG_ADDR *p6509Reg);
void P6509SetRcvrDefaults(P6509_RCVR_PARAMS *rcvr_params);
void P6509SetInputDefaults(P6509_INPUT_PARAMS *input_params);
void P6509SelectInputSource(P6509_RCVR_PARAMS *rcvr_params,
                            P6509_REG_ADDR *p6509Reg, unsigned int rcvr_mask);
void P6509SelectDataMode(P6509_RCVR_PARAMS *rcvr_params,
                         P6509_REG_ADDR *p6509Reg, unsigned int rcvr_mask);
void P6509SetInputSyncMode(P6509_INPUT_PARAMS *input_params,
                           P6509_REG_ADDR *p6509Reg, unsigned int input_mask);
void P6509SelectDataFormat(P6509_RCVR_PARAMS *rcvr_params,
                           P6509_REG_ADDR *p6509Reg, unsigned int rcvr_mask);
void P6509SelectTagEnable(P6509_RCVR_PARAMS *rcvr_params,
                          P6509_REG_ADDR *p6509Reg, unsigned int rcvr_mask);
void P6509SetCommPortToken(P6509_REG_ADDR *p6509Reg);
void P6509SetDataTransfer(P6509_RCVR_PARAMS *rcvr_params,
                          P6509_REG_ADDR *p6509Reg, unsigned int rcvr_mask);
void P6509ResetFifo(P6509_RCVR_PARAMS *rcvr_params, P6509_REG_ADDR *p6509Reg,
                    unsigned int rcvr_mask);
void P6509ReleaseFifo(P6509_RCVR_PARAMS *rcvr_params, P6509_REG_ADDR *p6509Reg,
                      unsigned int rcvr_mask);
void P6509SetIntrMask1(P6509_REG_ADDR *p6509Reg, unsigned long intr_mask);
void P6509SetIntrMask2(P6509_REG_ADDR *p6509Reg, unsigned long intr_mask);
void P6509ClearIntrMask1(P6509_REG_ADDR *p6509Reg, unsigned long intr_mask);
void P6509ClearIntrMask2(P6509_REG_ADDR *p6509Reg, unsigned long intr_mask);
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

A.1 **6509.h** (continued)

```
void P6509SetLo(P6509_RCVR_PARAMS *rcvr_params, P6509_REG_ADDR *p6509Reg,
                unsigned int rcvr_mask);
void P6509SetThresholdDetection(P6509_INPUT_PARAMS *input_params,
                                P6509_REG_ADDR *p6509Reg, unsigned int input_mask);
void P6509SetDecimationRate(P6509_RCVR_PARAMS *rcvr_params,
                            P6509_REG_ADDR *p6509Reg, unsigned int rcvr_mask);
void P6509SetDataMode(P6509_RCVR_PARAMS *rcvr_params, P6509_REG_ADDR *p6509Reg,
                      unsigned int rcvr_mask);
void P6509SetRcvrSyncMode(P6509_RCVR_PARAMS *rcvr_params,
                          P6509_REG_ADDR *p6509Reg, unsigned int rcvr_mask);
void P6509SetOutputMode(P6509_RCVR_PARAMS *rcvr_params,
                        P6509_REG_ADDR *p6509Reg, unsigned int rcvr_mask);
void P6509InitRcvr(P6509_RCVR_PARAMS *rcvr_params, P6509_REG_ADDR *p6509Reg,
                   unsigned int rcvr_mask);
int P6509DetermineGain(unsigned int dec_factor);
int *P6509ReadFifoCmplx(P6509_RCVR_PARAMS *rcvr_params, P6509_REG_ADDR *p6509Reg,
                        register volatile int *samples, register int num_samples,
                        unsigned int inputFifo);
int *P6509ReadCommPortCmplx(P6509_RCVR_PARAMS *rcvr_params,
                       register volatile int *samples,
                       register int num_samples, unsigned int inputCommPort);
                       void P6509UnpackCmplxData(P6509_RCVR_PARAMS *rcvr_params,
                       register volatile int *inSamples,
                       register volatile int *outSamples,
                       register int num_samples);
int *P6509ReadFifoReal(P6509_RCVR_PARAMS *rcvr_params, P6509_REG_ADDR *p6509Reg,
                       register volatile int *samples,
                       register int num_samples, unsigned int inputFifo);
int *P6509ReadCommPortReal(P6509_RCVR_PARAMS *rcvr_params,
                           register volatile int *samples,
                           register int num_samples,
                           unsigned int inputCommPort);
void P6509UnpackRealData(P6509_RCVR_PARAMS *rcvr_params,
                         register volatile int *inSamples,
                         register volatile int *outSamples,
                         register int num_samples);
#endif
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

A.2 init6509.c

```
/*************************************************************************
*
*   File : init6509.c
*
*   Description: This file contains routines for setting up the 6509.
*
*   $Revision:$   $Date:$
*
*   $History:$
*
*************************************************************************/

#include "math.h"
#include "6509.h"
#include "65xxcmd.h"
#ifdef FLOAT_64
#include "flotar.h"      /* Tartan's 64 bit Floating Point Library */
#endif
```

A.2.1 P6509InitRegAddr Function

```
/*************************************************************************
 Function: P6509InitRegAddr
 Description: This function initializes the P65XX_REG_ADDR structure with
              the addresses to each of the 6509 registers
 Inputs:
          regBaseAddr - Local address window to the 6509's Global Sram
          p6509Reg - Pointer to the P6509_REG_ADDR Table
 Outputs:
          Register Addresses stored in Address Table
 Returns:
 Notes:  Assumes a C30 or C40 is calling this routine
*************************************************************************/

void P6509InitRegAddr(unsigned long regBaseAddr, P6509_REG_ADDR *p6509Reg)
{
    p6509Reg->Int_Mask_1 = (unsigned long *)regBaseAddr +
                                            (P6509_INTR_MASK_REG1 / 4);
    p6509Reg->Int_Mask_2 = (unsigned long *)regBaseAddr +
                                            (P6509_INTR_MASK_REG2 / 4);
    p6509Reg->Stat_Reg_1 = (unsigned long *)regBaseAddr +
                                            (P6509_STATUS_REG1 / 4);
    p6509Reg->Stat_Reg_2 = (unsigned long *)regBaseAddr +
                                            (P6509_STATUS_REG2 / 4);
    p6509Reg->Fifo_Reset = (unsigned long *)regBaseAddr +
                                            (P6509_FIFO_RESET_REG / 4);
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

A.2 **init6509.c** (continued)

A.2.1 P6509InitRegAddr Function (continued)

```
    p6509Reg->Out_Comm_Ctl = (unsigned long *)regBaseAddr +
                                              (P6509_COMM_PORT_CNTL_REG / 4);
    p6509Reg->Gray_1_Base = (unsigned long *)regBaseAddr +
                                              (P6509_GRAY_1_BASE / 4);
    p6509Reg->Mode_Slct = (unsigned long *)regBaseAddr +
                                              (P6509_MODE_SLCT_REG / 4);
    p6509Reg->Tag_En = (unsigned long *)regBaseAddr +
                                              (P6509_TAG_ENABLE_REG / 4);
    p6509Reg->In_Mux = (unsigned long *)regBaseAddr +
                                              (P6509_INPUT_MUX_REG / 4);
    p6509Reg->Thrshld_Detect_A = (unsigned long *)regBaseAddr +
                                              (P6509_THRSHLD_DETECT_A / 4);
    p6509Reg->Thrshld_Detect_B = (unsigned long *)regBaseAddr +
                                              (P6509_THRSHLD_DETECT_B / 4);
    p6509Reg->Fifo = (unsigned long *)regBaseAddr +
                                              (P6509_RCVR_1_FIFO / 4);
    p6509Reg->VME_Intr_Reg = (unsigned long *)regBaseAddr +
                                          (P6509_VME_INTR_REG / 4);
    p6509Reg->Intr_Ackn_Addr = (unsigned long *)regBaseAddr +
                                          (P6509_INTR_ACKN_REG / 4);
}
```

A.2.2 P6509ResetRegs Function

```
/*****************************************************************************
 Function: P6509ResetRegs
 Description: This function initializes the 6509 registers to a known state
 Inputs:
         p6509Reg - Pointer to the 6509's address table
 Outputs:
 Returns:
 Notes:
*****************************************************************************/

void P6509ResetRegs(P6509_REG_ADDR *p6509Reg)
{
    *p6509Reg->Int_Mask_1 = 0;
    *p6509Reg->Int_Mask_2 = 0;
    *p6509Reg->Fifo_Reset = 0x100;
    *p6509Reg->Out_Comm_Ctl = 0x1111;
    *p6509Reg->Mode_Slct = P6509_SW_AS_OFF | P6509_SW_SS_OFF |
                           P6509_SW_SYNC_SRC;
    *p6509Reg->Tag_En = P6509_SW_AS_OFF | P6509_SW_SS_OFF | P6509_SW_SYNC_SRC;
    *p6509Reg->In_Mux = 0;
    *p6509Reg->Thrshld_Detect_A = 0;
    *p6509Reg->Thrshld_Detect_B = 0;
}
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

A.2 **init6509.c** (continued)

A.2.3 P6509InitRcvrRegs

```
/*****************************************************************************
 Function: P6509InitRcvrRegs
 Description:  This function initializes the 6509 Receivers to the settings
               specified
 Inputs:
        rcvr_params - Pointer to the P65XX_RCVR_PARAMS structure containing
                      the settings
        p6509Reg - Pointer to the 6509's address table
        rcvr_mask - Mask specifying which receviers are affected
 Outputs:
 Returns:
 Notes:
*****************************************************************************/

void P6509InitRcvrRegs(P6509_RCVR_PARAMS *rcvr_params, P6509_REG_ADDR *p6509Reg,
                       unsigned int rcvr_mask)
{
    unsigned long temp;
    /* Select Input Source */
    P6509SelectInputSource(rcvr_params, p6509Reg, rcvr_mask);
    /* Set Data Mode */
    P6509SelectDataMode(rcvr_params, p6509Reg, rcvr_mask);
    /* Select Data Format */
    P6509SelectDataFormat(rcvr_params, p6509Reg, rcvr_mask);
    /* Select Tagging */
    P6509SelectTagEnable(rcvr_params, p6509Reg, rcvr_mask);
    /* Set Data Transfer Method */
    P6509SetDataTransfer(rcvr_params, p6509Reg, rcvr_mask);

    /* Initialize Receiver */
    P6509InitRcvr(rcvr_params, p6509Reg, rcvr_mask);
}
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

A.2 **init6509.c** (continued)

A.2.4 P6509InitInputRegs

```
/*****************************************************************************
 Function: P6509InitInputRegs
 Description:  This function initializes the 6509 Input Channels to the settings
               specified
 Inputs:   input_params - Pointer to the P65XX_INPUT_PARAMS structure containing
                         the input channel settings
           p6509Reg - Pointer to the 6509's address table
           input_mask - Mask specifying which input channels are affected
 Outputs:
 Returns:
 Notes:
*****************************************************************************/

void P6509InitInputRegs(P6509_INPUT_PARAMS *input_params,
                        P6509_REG_ADDR *p6509Reg,
                        unsigned int input_mask)
{
    /* Set Threshold Detection */
    P6509SetThresholdDetection(input_params, p6509Reg, input_mask);

    /* Set Input Sync Mode */
    P6509SetInputSyncMode(input_params, p6509Reg, input_mask);
}
```

A.2.5 P6509SetRcvrDefaults Function

```
/*****************************************************************************
 Function: P6509SetRcvrDefaults
 Description:  This function places default settings in the receiver parameter
               structure.
 Inputs:   rcvr_params - Pointer to the P65XX_RCVR_PARAMS structure being initialized
 Outputs:
 Returns:
 Notes:
*****************************************************************************/

void P6509SetRcvrDefaults(P6509_RCVR_PARAMS *rcvr_params)
{
    rcvr_params->decimation_rate = 64;
    rcvr_params->data_mode = P65XX_COMPLEX;
    rcvr_params->data_source = P65XX_INPUT_A;
    rcvr_params->packing_mode = P65XX_PACKED;
    rcvr_params->data_order = P65XX_IQ_ORDER;
    rcvr_params->data_transfer = P65XX_FIFO;
    rcvr_params->rcvr_select = P65XX_ODD_RECEIVER;
    rcvr_params->tag_enable = P65XX_TAG_OFF;
    rcvr_params->flip_spectrum = P65XX_FLIP_OFF;
    rcvr_params->offset_spectrum = P65XX_OFFSET_ON;
    rcvr_params->filter_select = P6509_80_PERCENT;
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

A.2 **init6509.c** (continued)

A.2.5 **P6509SetRcvrDefaults** (continued)

```
    rcvr_params->rcvr_sync_mode = P6509_GC_LD_FREQ | P6509_GC_SS_OFF;
    rcvr_params->lo = 3.0e6;
    rcvr_params->input_clock_rate = 70.0e6;
}
```

A.2.6 P6509SetInputDefaults Function

```
/*************************************************************************
 Function: P6509SetInputDefaults
 Description:  This function places default settings in the input channel
               parameter structure.
 Inputs:   input_params -   Pointer to the P65XX_INPUT_PARAMS structure being
                            initialized.
 Outputs:
 Returns:
 Notes:
*************************************************************************/

void P6509SetInputDefaults(P6509_INPUT_PARAMS *input_params)
{
    input_params->threshold_level = P65XX_THRSHLD_DETECT_OFF;
    input_params->threshold_cross_cnt = P65XX_CROSS_CNT_2;
    input_params->input_sync_mode = 0;
}
```

A.2.7 P6509SelectInputSource Function

```
/*************************************************************************
 Function: P6509SelectInputSource
 Description:  This function selects the input source for the receiver.
 Inputs:   rcvr_params - Pointer to the P65XX_RCVR_PARAMS structure containing
                         the settings
           p6509Reg - Pointer to the 6509's address table
           rcvr_mask - Mask specifying which receviers are affected
 Outputs:
 Returns:
 Notes:
*************************************************************************/

void P6509SelectInputSource(P6509_RCVR_PARAMS *rcvr_params,
                            P6509_REG_ADDR *p6509Reg, unsigned int rcvr_mask)
{
    if (rcvr_params->data_source == P65XX_INPUT_A)
    {
           *p6509Reg->In_Mux = *p6509Reg->In_Mux & ~(rcvr_mask << 8);
    }
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

A.2 **init6509.c** (continued)

A.2.7 **P6509SelectInputSource** (continued)

```
   else /* data_source == P65XX_INPUT_B */
    {
           *p6509Reg->In_Mux = *p6509Reg->In_Mux | (rcvr_mask << 8);
    }
}
```

A.2.8 P6509SelectDataMode Function

```
/*****************************************************************************
 Function: P6509SelectDataMode
 Description:  This function sets the data mode for a receiver as Real or
               Complex.
 Inputs:
        rcvr_params - Pointer to the P65XX_RCVR_PARAMS structure containing
                      the settings
        p6509Reg - Pointer to the 6509's address table
        rcvr_mask - Mask specifying which receviers are affected
 Outputs:
 Returns:
 Notes:
*****************************************************************************/

void P6509SelectDataMode(P6509_RCVR_PARAMS *rcvr_params,
                         P6509_REG_ADDR *p6509Reg, unsigned int rcvr_mask)
{
    if (rcvr_params->data_mode == P65XX_REAL)
           *p6509Reg->Mode_Slct = (*p6509Reg->Mode_Slct & ~(rcvr_mask << 8)) |
                                  (rcvr_mask << 8);
    else   /* data_mode == P65XX_COMPLEX */
           *p6509Reg->Mode_Slct = (*p6509Reg->Mode_Slct & ~(rcvr_mask << 8));
}
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

A.2 **init6509.c** (continued)

A.2.9 P6509SetInputSyncMode

```
/*************************************************************************
 Function: P6509SetInputSyncMode
 Description:  This function sets the input sync mode for each of the input
               channels.
 Inputs:   input_params -  Pointer to the P65XX_INPUT_PARAMS structure containing
                          the settings
           p6509Reg - Pointer to the 6509's address table
           input_mask - Mask specifying which input channels are affected
 Outputs:
 Returns:
 Notes:
*************************************************************************/

void P6509SetInputSyncMode(P6509_INPUT_PARAMS *input_params,
                           P6509_REG_ADDR *p6509Reg, unsigned int input_mask)
{
    volatile unsigned long *cntl = p6509Reg->Mode_Slct;
    unsigned int i;

    for (i = 0; i < P6509_NUM_INPUT_CHANNELS; ++i)
    {
           if (input_mask & (1 << i))
             *cntl = (*cntl & ~0xf) | input_params->input_sync_mode;
        cntl = p6509Reg->Tag_En;
    }
}
```

A.2.10 P6509SelectTagEnable Function

```
/*************************************************************************
 Function: P6509SelectTagEnable
 Description:  This function enables/disables tagging of the output data.
 Inputs:   rcvr_params - Pointer to the P65XX_RCVR_PARAMS structure containing
                         the settings
           p6509Reg - Pointer to the 6509's address table
           rcvr_mask - Mask specifying which receviers are affected
 Outputs:
 Returns:
 Notes:
*************************************************************************/

void P6509SelectTagEnable(P6509_RCVR_PARAMS *rcvr_params,
                          P6509_REG_ADDR *p6509Reg, unsigned int rcvr_mask)
{
    if (rcvr_params->tag_enable == P65XX_TAG_ON)
           *p6509Reg->Tag_En = (*p6509Reg->Tag_En & ~(rcvr_mask << 8)) |
                               (rcvr_mask << 8);
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

A.2 **init6509.c** (continued)

A.2.10 P6509SelectTagEnable Function (continued)

```
    else    /* tag_enable == P65XX_TAG_OFF */
            *p6509Reg->Tag_En = (*p6509Reg->Tag_En & ~(rcvr_mask << 8));
}
```

A.2.11 P6509SelectDataFormat Function

```
/*****************************************************************************
 Function: P6509SelectDataFormat
 Description:  This function selects the format of the data output to the
               front panel communication ports.
 Inputs:   rcvr_params - Pointer to the P65XX_RCVR_PARAMS structure containing
                        the settings
           p6509Reg - Pointer to the 6509's address table
           rcvr_mask - Mask specifying which receviers are affected
 Outputs:
 Returns:
 Notes:
*****************************************************************************/

void P6509SelectDataFormat(P6509_RCVR_PARAMS *rcvr_params,
                           P6509_REG_ADDR *p6509Reg, unsigned int rcvr_mask)
{
     unsigned long temp_data = 0;
     unsigned long data_word = 0;
     unsigned long out_mask = 0xffff;
     unsigned int  i;

     if (rcvr_params->packing_mode == P65XX_PACKED)
            temp_data = 0x8;
     if ((rcvr_params->data_order == P65XX_IQ_ORDER) ||
         (rcvr_params->data_order == P65XX_TIME_ORDER))
            temp_data |= 0x4;
     if (rcvr_params->rcvr_select == P65XX_ODD_RECEIVER)
            temp_data |= 0x2;
     for (i = 0; i < P6509_NUM_COMM_PORTS; ++i)
     {
            if (rcvr_mask & (3 << (i * 2)))
         {
                data_word |= temp_data << (i * 4);
             out_mask = (out_mask & ~(0xf << (i * 4))) | (1 << (i * 4));
         }
     }
     *p6509Reg->Out_Comm_Ctl = (*p6509Reg->Out_Comm_Ctl & out_mask) |
                               data_word;
}
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

A.2 **init6509.c** (continued)

A.2.12 P6509SetCommPortToken Function

```
/*****************************************************************************
 Function: P6509SetCommPortToken
 Description:  This function initializes the input communication ports to the
               C40.  This function makes the 4 front panel communication ports
               input ports.
 Inputs:       p6509Reg - Pointer to the 6509's address table
 Outputs:
 Returns:
 Notes:  This function doesn't affect the four output ports. This function can be
         called after the processor has been reset one time ONLY.
*****************************************************************************/

void P6509SetCommPortToken(P6509_REG_ADDR *p6509Reg)
{
     *p6509Reg->Fifo_Reset &= ~0x100;
     *p6509Reg->Fifo_Reset |= 0x100;
}
```

A.2.13 P6509SetDataTransfer Function

```
/*****************************************************************************
 Function: P6509SetDataTransfer
 Description:  This function sets the output source for reading the data.
 Inputs:  rcvr_params - Pointer to the P65XX_RCVR_PARAMS structure containing
                        the settings
          p6509Reg - Pointer to the 6509's address table
          rcvr_mask - Mask specifying which receviers are affected
 Outputs:
 Returns:
 Notes:
*****************************************************************************/

void P6509SetDataTransfer(P6509_RCVR_PARAMS *rcvr_params,
                          P6509_REG_ADDR *p6509Reg, unsigned int rcvr_mask)
{
     int           i;
     unsigned long data_word = 0;
     unsigned long temp_data = 0;
     unsigned long out_mask = 0xffff;
     if (rcvr_params->data_transfer == P65XX_FIFO)
            temp_data = 0x1;

     for (i = 0; i < P6509_NUM_COMM_PORTS; ++i)
     {
             if (rcvr_mask & (3 << (i * 2)))
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

A.2 **init6509.c** (continued)

A.2.13 P6509SetDataTransfer Function (continued)

```
            {
                   data_word |= temp_data << (i * 4);
                out_mask = (out_mask & ~(0xf << (i * 4))) | (0xE << (i * 4));
            }
        }
        *p6509Reg->Out_Comm_Ctl = (*p6509Reg->Out_Comm_Ctl & out_mask) |
                                  data_word;
}
```

A.2.14 P6509ResetFifo Function

```
/*****************************************************************************
 Function: P6509ResetFifo
 Description:  This function puts the fifo in a reset state for the given
               receivers.
 Inputs:   rcvr_params - Pointer to the P65XX_RCVR_PARAMS structure containing
                         the settings
           p6509Reg - Pointer to the 6509's address table
           rcvr_mask - Mask specifying which receviers are affected
 Outputs:
 Returns:
 Notes:    When reading data directly from the fifos, each fifo can be reset
           indiviually. When reading from Comm Ports, Fifos must be reset in  pairs.
*****************************************************************************/

void P6509ResetFifo(P6509_RCVR_PARAMS *rcvr_params, P6509_REG_ADDR *p6509Reg,
                    unsigned int rcvr_mask)
{
    unsigned long i;
    unsigned long reset_word = *p6509Reg->Fifo_Reset;

    /* In VME Mode, all fifos can be reset individually.  In Comm Port Mode, */
    /* the fifos have to be reset in pairs(1&2, 3&4, 5&6, 7&8).               */

    if (rcvr_params->data_transfer == P65XX_FIFO)
    {
        *p6509Reg->Fifo_Reset = *p6509Reg->Fifo_Reset & ~rcvr_mask;
    }
    else   /* P6509_COMM_PORT */
    {
        for (i = 0; i < P6509_NUM_RCVRS; i+=2)
        {
               if (rcvr_mask & (0x3 << i
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

A.2 **init6509.c** (continued)

A.2.14 P6509SetDataTransfer Function (continued)

```
                {
                  reset_word = reset_word & ~(0x3 << i);
                }
            }
            *p6509Reg->Fifo_Reset = reset_word;
        }
    }
```

A.2.15 P6509ReleaseFifo Function

```
/*****************************************************************************
 Function: P6509ReleaseFifo
 Description:  This function release the fifo(s) for the given receivers.
 Inputs:  rcvr_params - Pointer to the P65XX_RCVR_PARAMS structure containing
                        the settings
          p6509Reg - Pointer to the 6509's address table
          rcvr_mask - Mask specifying which receviers are affected
 Outputs:
 Returns:
 Notes:  When reading data directly from the fifos, each fifo can be released
         indiviually. When reading from Comm Ports, Fifos must be released in  pairs.
*****************************************************************************/

void P6509ReleaseFifo(P6509_RCVR_PARAMS *rcvr_params, P6509_REG_ADDR *p6509Reg,
                      unsigned int rcvr_mask)
{
    unsigned long i;
    unsigned long reset_word = *p6509Reg->Fifo_Reset;

    /* In VME Mode, all fifos can be released individually. In Comm Port   */
    /* Mode, the fifos have to be reset in pairs(1&2, 3&4, 5&6, 7&8).      */

    if (rcvr_params->data_transfer == P65XX_FIFO)
    {
        *p6509Reg->Fifo_Reset = *p6509Reg->Fifo_Reset | rcvr_mask;
    }
    else   /* P6509_COMM_PORT */
    {
        for (i = 0; i < P6509_NUM_RCVRS; i+=2)
        {
               if (rcvr_mask & (0x3 << i))
               {
                 reset_word = reset_word | (0x3 << i);
            }
        }
        *p6509Reg->Fifo_Reset = reset_word;
    }
}
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

A.2 **init6509.c** (continued)

A.2.16 P6509SetThresholdDetection Function

```
/*************************************************************************
 Function: P6509SetThresholdDetection
 Description:  This function sets up threshold detection on a specified
               input channel.
 Inputs:   input_params - Pointer to the P65XX_INPUT_PARAMS structure containing
                         the input channel settings
           p6509Reg - Pointer to the 6509's address table
           input_mask - Mask specifying which input channels are affected
 Outputs:
 Returns:
 Notes:
*************************************************************************/

void P6509SetThresholdDetection(P6509_INPUT_PARAMS *input_params,
                                P6509_REG_ADDR *p6509Reg,
                                unsigned int input_mask)
{
    unsigned int i;
    volatile unsigned long *thrshld_detect = p6509Reg->Thrshld_Detect_A;
    for (i = 0; i < P6509_NUM_INPUT_CHANNELS; i++)
    {
        if (input_mask & (1 << i))
        {
            *thrshld_detect = input_params->threshold_level |
                              input_params->threshold_cross_cnt;
        }
        thrshld_detect = p6509Reg->Thrshld_Detect_B;
    }
}
```

A.2.17 P6509SetIntrMask1

```
/*************************************************************************
 Function: P6509SetIntrMask1
 Description: This function enables the fifo full, clock not ready and threshold
              detection interrupts.
 Inputs:   p6509Reg - Pointer to the 6509's address table
           intr_mask - Mask containing interrupts to be enabled
 Outputs:
 Returns:
 Notes:
*************************************************************************/

void P6509SetIntrMask1(P6509_REG_ADDR *p6509Reg, unsigned long intr_mask)
{
    unsigned long intr_cntl = (intr_mask & 0x01ff0000) >> 16;

    if (intr_mask & P65XX_CLOCK_NOT_READY_B)
        intr_cntl |= P6509_IM1_CLOCK_NOT_READY_B;
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

A.2 **init6509.c** (continued)

A.2.17 P6509SetIntrMask Function (continued)

```
    if (intr_mask & P65XX_THRSHLD_DETECT_A)
        intr_cntl |= P6509_IM1_THRSHLD_DETECT_A;
    if (intr_mask & P65XX_THRSHLD_DETECT_B)
        intr_cntl |= P6509_IM1_THRSHLD_DETECT_B;

    *p6509Reg->Int_Mask_1 |= intr_cntl;
}
```

A.2.18 P6509ClearIntrMask1 Function

```
/*****************************************************************************
 Function: P6509ClearIntrMask1
 Description: This function disables the fifo full, clock not ready and threshold
              detection interrupts.
 Inputs:   p6509Reg - Pointer to the 6509's address table
           intr_mask - Mask containing interrupts to be disabled
 Outputs:
 Returns:
 Notes:
*****************************************************************************/

void P6509ClearIntrMask1(P6509_REG_ADDR *p6509Reg, unsigned long intr_mask)
{
    unsigned long intr_cntl = (intr_mask & 0x01ff0000) >> 16;
    if (intr_mask & P65XX_CLOCK_NOT_READY_B)
        intr_cntl |= P6509_IM1_CLOCK_NOT_READY_B;

    if (intr_mask & P65XX_THRSHLD_DETECT_A)
        intr_cntl |= P6509_IM1_THRSHLD_DETECT_A;
    if (intr_mask & P65XX_THRSHLD_DETECT_B)
        intr_cntl |= P6509_IM1_THRSHLD_DETECT_B;

    *p6509Reg->Int_Mask_1 &= ~intr_cntl;
}
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

A.2 **init6509.c** (continued)

A.2.19 P6509SetIntrMask2 Function

```
/*************************************************************************
 Function: P6509SetIntrMask2
 Description: This function enables the fifo half full and fifo not empty
              interrupts.
 Inputs:   p6509Reg - Pointer to the 6509's address table
           intr_mask - Mask containing interrupts to be enabled
 Outputs:
 Returns:
 Notes:
*************************************************************************/

void P6509SetIntrMask2(P6509_REG_ADDR *p6509Reg, unsigned long intr_mask)
{
    *p6509Reg->Int_Mask_2 |= intr_mask & 0xffffL;
}
```

A.2.20 P6509ClearIntrMask2 Function

```
/*************************************************************************
 Function: P6509ClearIntrMask2
 Description: This function disables the fifo half full and fifo not empty
              interrupts.
 Inputs:   6509Reg - Pointer to the 6509's address table
           intr_mask - Mask containing interrupts to be disabled
 Outputs:
 Returns:
 Notes:
*************************************************************************/

void P6509ClearIntrMask2(P6509_REG_ADDR *p6509Reg, unsigned long intr_mask)
{
    *p6509Reg->Int_Mask_2 &= ~intr_mask & 0xffffL;
}
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

A.2 **init6509.c** (continued)

A.2.21 P6509SetLO

```
/*****************************************************************************
 Function: P6509SetLO
 Description: This function sets the Local Oscillator Frequency for the
              specified receiver
 Inputs:   rcvr_params - Pointer to the P65XX_RCVR_PARAMS structure containing
                         the reciever's settings
           p6509Reg - Pointer to the 6509's address table
           rcvr_mask - Mask specifying which receviers are affected
 Outputs:
 Returns:
 Notes:  64 Bit floating point provides a more accurate LO Setting
*****************************************************************************/

void P6509SetLo(P6509_RCVR_PARAMS *rcvr_params, P6509_REG_ADDR *p6509Reg,
                unsigned int rcvr_mask)
{
    unsigned int i;
    volatile unsigned long *gray;
    unsigned int freq;
#ifndef FLOAT_64
    float        temp;
    temp = (pow(2.0, 28.0) / rcvr_params->input_clock_rate) *
           (rcvr_params->lo / 2.0);
    freq = (unsigned int) temp * 2;
#else
    float64  f64temp;
    float64  f64temp1 = {0x00000000, 2.68435456e+08};
    divf64f32(&f64temp, &f64temp1, rcvr_params->input_clock_rate);
    mpyf64f32(&f64temp1, &f64temp, rcvr_params->lo);
    freq = roundu32f64(&f64temp1);
#endif
    for (i = 0; i < P6509_NUM_RCVRS; ++i)
    {
           if (rcvr_mask & (1 << i))
           {
               gray = (unsigned long *)(p6509Reg->Gray_1_Base + 0x10 * i);
               gray[P6509_CH_FREQ_BYTE_0] = freq & 0x000000ff;
               gray[P6509_CH_FREQ_BYTE_1] = (freq >> 8) & 0x000000ff;
               gray[P6509_CH_FREQ_BYTE_2] = (freq >> 16) & 0x000000ff;
               gray[P6509_CH_FREQ_BYTE_3] = (freq >> 24) & 0x000000ff;
        }
    }
}
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

A.2 **init6509.c** (continued)

A.2.22 P6509SetDecimationRate Function

```
/*****************************************************************************
 Function: P6509SetDecimationRate
 Description: This function sets the decimation rate for the specified receiver
 Inputs:   rcvr_params - Pointer to the P65XX_RCVR_PARAMS structure containing
                         the reciever's settings
           p6509Reg - Pointer to the 6509's address table
           rcvr_mask - Mask specifying which receviers are affected
 Outputs:
 Returns:
 Notes:
*****************************************************************************/

void P6509SetDecimationRate(P6509_RCVR_PARAMS *rcvr_params,
                            P6509_REG_ADDR *p6509Reg, unsigned int rcvr_mask)
{
    unsigned int i;
    unsigned long dec_factor;
    volatile unsigned long *gray;

    /* Adjust Decimation Rate by 4D */

    if (rcvr_params->data_mode == P65XX_COMPLEX)
        dec_factor = (rcvr_params->decimation_rate / 4) - 1;
    else
        dec_factor = (rcvr_params->decimation_rate / 2) - 1;
    for (i = 0; i < P6509_NUM_RCVRS; ++i)
    {
           if (rcvr_mask & (1 << i))
           {
               gray = (unsigned long *)(p6509Reg->Gray_1_Base + 0x10 * i);
            /* Set gain based on decimation rate */
               gray[P6509_CH_GAIN_CONTROL] =
                         (int) P6509DetermineGain(rcvr_params->decimation_rate);
            /* Set Decimation Rate */
               gray[P6509_CH_DECIMATION_LSB] = dec_factor & 0xff;
               gray[P6509_CH_DECIMATION_MSB] = (dec_factor >> 8) & 0xff;
        }
    }
}
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

A.2 **init6509.c** (continued)

A.2.23 P6509SetDataMode Function

```
/*************************************************************************
 Function: P6509SetDataMode
 Description: This function sets up the receiver for the selected data mode,
              flip and offset specified.
 Inputs:   rcvr_params - Pointer to the P65XX_RCVR_PARAMS structure containing
                         the settings
           p6509Reg - Pointer to the 6509's address table
           rcvr_mask - Mask specifying which receviers are affected
 Outputs:
 Returns:
 Notes:
*************************************************************************/

void P6509SetDataMode(P6509_RCVR_PARAMS *rcvr_params, P6509_REG_ADDR *p6509Reg,
                      unsigned int rcvr_mask)
{
    unsigned int i;
    volatile unsigned long *gray;
    unsigned long output_status;
    unsigned long filter_mode;
    filter_mode = rcvr_params->filter_select;
    if (rcvr_params->flip_spectrum)
        filter_mode |= 0x8;
    if (rcvr_params->offset_spectrum)
        filter_mode |= 0x4;
    if (rcvr_params->data_mode == P65XX_COMPLEX)
        output_status = 0x0;
    else
    {
        filter_mode |= 0x2;
        output_status = 0x80;
    }
    for (i = 0; i < P6509_NUM_RCVRS; ++i)
    {
          if (rcvr_mask & (1 << i))
          {
              gray = (unsigned long *)(p6509Reg->Gray_1_Base + 0x10 * i);
              gray[P6509_CH_FILTER_MODE] = filter_mode;
              gray[P6509_CH_OUTPUT_STATUS] = output_status;
        }
    }
}
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

A.2 **init6509.c** (continued)

A.2.24 P6509SetRcvrSyncMode Function

```
/*************************************************************************
 Function: P6509SetRcvrSyncMode
 Description:  This function sets the Graychip's sync mode register.
 Inputs:   rcvr_params - Pointer to the P65XX_RCVR_PARAMS structure containing
                         the settings
           p6509Reg - Pointer to the 6509's address table
           rcvr_mask - Mask specifying which receviers are affected
 Outputs:
 Returns:
 Notes:
*************************************************************************/

void P6509SetRcvrSyncMode(P6509_RCVR_PARAMS *rcvr_params,
                          P6509_REG_ADDR *p6509Reg, unsigned int rcvr_mask)
{
    unsigned int i;
    unsigned long dec_factor;
    volatile unsigned long *gray;
    for (i = 0; i < P6509_NUM_RCVRS; ++i)
    {
           if (rcvr_mask & (1 << i))
           {
               gray = (unsigned long *)(p6509Reg->Gray_1_Base + 0x10 * i);
               gray[P6509_CH_SYNC_MODE] = rcvr_params->rcvr_sync_mode;
        }
    }
}
```

A.2.25 P6509SetOutputMode Function

```
/*************************************************************************
 Function: P6509SetOutputMode
 Description:  This function Sets the Graychip's Output Mode register.
 Inputs:   rcvr_params - Pointer to the P65XX_RCVR_PARAMS structure containing
                        the settings
           p6509Reg - Pointer to the 6509's address table
           rcvr_mask - Mask specifying which receviers are affected
 Outputs:
 Returns:
 Notes:
*************************************************************************/

void P6509SetOutputMode(P6509_RCVR_PARAMS *rcvr_params,
                        P6509_REG_ADDR *p6509Reg, unsigned int rcvr_mask)
{
    unsigned int i;
    unsigned long dec_factor;
    volatile unsigned long *gray;
    for (i = 0; i < P6509_NUM_RCVRS; ++i)
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

A.2 **init6509.c** (continued)

A.2.25 **P6509SetOutputMode** (continued)

```
    {
            if (rcvr_mask & (1 << i))
            {
                gray = (unsigned long *)(p6509Reg->Gray_1_Base + 0x10 * i);
                gray[P6509_CH_OUTPUT_MODE] = 0x52; /* quarter rate, normal BS */
        }
    }
}
```

A.2.26 **P6509InitRcvr Function**

```
/*****************************************************************************
 Function: P6509InitRcvr
 Description:  This function initializes the all the Graychip registers
               with the current settings.
 Inputs:   rcvr_params - Pointer to the P65XX_RCVR_PARAMS structure containing
                         the settings
           p6509Reg - Pointer to the 6509's address table
           rcvr_mask - Mask specifying which receviers are affected
 Outputs:
 Returns:
 Notes:
*****************************************************************************/

void P6509InitRcvr(P6509_RCVR_PARAMS *rcvr_params, P6509_REG_ADDR *p6509Reg,
                   unsigned int rcvr_mask)
{

    /* Set Tuning Frequency */

    P6509SetLo(rcvr_params, p6509Reg, rcvr_mask);

    /* Set Decimation Rate */

    P6509SetDecimationRate(rcvr_params, p6509Reg, rcvr_mask);

    /* Set Data Mode */

    P6509SetDataMode(rcvr_params, p6509Reg, rcvr_mask);
    /* Set Sync Mode */

    P6509SetRcvrSyncMode(rcvr_params, p6509Reg, rcvr_mask);
    /* Set Output Mode */

    P6509SetOutputMode(rcvr_params, p6509Reg, rcvr_mask);
}
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

A.2 **init6509.c** (continued)

A.2.27 P6509DetermineGain Function

```
/*****************************************************************************
 Function: P6509DetermineGain
 Description: This function determines the gain setting for the graychip
              based on the decimation factor.
 Inputs:   dec_factor - Decimation Factor
 Outputs:  gain - Gain setting
 Returns:
 Notes:
*****************************************************************************/

P6509DetermineGain(unsigned int dec_factor)
{
    double fraction;
    int    power;
    int    gain;
    fraction = frexp((double)dec_factor, &power);
    fraction *= 2.0;
    if (fraction != 1.0)
    {
       fraction = (float) dec_factor / pow(2.0, (float) power);
       gain = (int)((80.0 * log10(fraction)) / 6.0);
    }
    else    /* Unity Gain */
          gain = 0;
    return(abs(gain));
}
```

A.2.28 P6509ReadFifoCmplx Function

```
/*****************************************************************************
 Function: P6509ReadFifoCmplx
 Description:  This function reads complex data from the fifo.
 Inputs:   rcvr_params - Pointer to the P65XX_RCVR_PARAMS structure containing
                         the receiver settings
           p6509Reg - Pointer to the 6509's address table
           samples - Pointer to a data buffer to store data
           num_samples - Number of samples to be read
           inputFifo - Fifo to be read (1 - 8)
 Outputs:  Data is stored to the data buffer in the following format:
              Real,Imag,Real,Imag......
 Returns:  samples - Address of the next available location in data buffer
 Notes:    Data is sign extended and right justified
*****************************************************************************/

int *P6509ReadFifoCmplx(P6509_RCVR_PARAMS *rcvr_params,
                        P6509_REG_ADDR *p6509Reg,
                        register volatile int *samples,
                        register int num_samples, unsigned int inputFifo)
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

A.2 **init6509.c** (continued)

A.2.28 **P6509ReadFifoCmplx Function** (continued)

```
{
    int          num_samples_to_read;
    register volatile int *fifo = (int *)p6509Reg->Fifo +
                                  ((0x8000 * (inputFifo - 1)) / 4);
    register int temp;
    register int temp1;
    register int i;
    while (num_samples > 0)
    {
        /* wait FIFO half-full */
           while ((*p6509Reg->Stat_Reg_2 & (0x100 << (inputFifo-1))) != 0);
           /* Determine how many samples left to read */
           if (num_samples <= P6509_FIFO_SIZE/2)
            num_samples_to_read = num_samples;
           else
               num_samples_to_read = P6509_FIFO_SIZE/2;

           /* Read out data from the fifo */

           for (i = 0; i < num_samples_to_read; i++)
           {
               temp = *fifo;    /* store real and imaginary */
               temp1 = *fifo;
               *samples++ = temp1 >> 16;    /* Real(I) Value            */
               *samples++ = temp >> 16;     /* Imag(Q) Value            */
           }
           /* Decrement number of samples needed */
           num_samples -= num_samples_to_read;
    }
    return((int *)samples);
}
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

A.2 **init6509.c** (continued)

A.2.29 P6509ReadCommPortCmplx Function

```
/*************************************************************************
 Function: P6509ReadCommPortCmplx
 Description:  This function reads complex data from the Comm Port.
 Inputs:   rcvr_params - Pointer to the P65XX_RCVR_PARAMS structure containing
                        the receiver settings
           samples - Pointer to a data buffer to store data
           num_samples - Number of samples to be read
           inputCommPort - Comm Port to be read (0 - 5)
 Outputs:  Data is stored to the data buffer in the following format:
              Unpacked Data -  Real,Imag,Real,Imag ....  Sign Extended
                               Right Justified
              Packed, IQ Ordered - Real/Imag, Real/Imag ....
              Packed, Receiver Ordered - Imag(Even)/Imag(Odd),Real(Even)/Real(Odd).
Returns:  samples - Address of the next available location in data buffer
Notes:
*************************************************************************/

int *P6509ReadCommPortCmplx(P6509_RCVR_PARAMS *rcvr_params,
                            register volatile int *samples,
                            register int num_samples,
                            unsigned int inputCommPort)
{
    int num_samples_to_read;
    int temp;
    register int i;
    volatile unsigned int *comStatus = (unsigned int *) 0x100040 +
          0x10 * inputCommPort;
    register volatile int *comInput = (int *)comStatus + 1;

    /* Read out data from the fifo */

    for (i = 0; i < num_samples; i++)
    {
          while ((*comStatus & 0x1e00) == 0);
        if (rcvr_params->packing_mode == P65XX_UNPACKED)
        {
            temp = *comInput;
            *samples++ = *comInput >> 16;   /* Real(I) Value            */
            *samples++ = temp >> 16;        /* Imag(Q) Value            */
        }
        else if (rcvr_params->data_order == P65XX_IQ_ORDER)
        {
              *samples++ = *comInput;   /* Real(I)/Imag(Q) */
        }
        else if (rcvr_params->data_order == P65XX_RECEIVER_ORDER)
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

A.2 **init6509.c** (continued)

A.2.29 P6509ReadCommPortCmplx Function (continued)

```
        {
                *samples++ = *comInput;    /* Imag(Q) Even/Imag(Q) Odd */
                *samples++ = *comInput;    /* Real(I) Even/Real(I) Odd */
        }
    }
    return((int *)samples);
}
```

A.2.30 P6509UnpackCmplxData Function

```
/*****************************************************************************
 Function: P6509UnpackCmplxData
 Description:  This function unpacks a complex data buffer read from a Comm Port
               based on the packing mode selected.
 Inputs:   rcvr_params - Pointer to the P65XX_RCVR_PARAMS structure containing
                        the receiver settings
           inSamples - Pointer to a data buffer containing complex data
           outSamples - Pointer to the output data buffer
           num_samples - Number of samples to be read
 Outputs:  Data is stored to the data buffer in the following format:
              Real,Imag,Real,Imag......
 Returns:
 Notes:    Data is sign extended and right justified
*****************************************************************************/

void P6509UnpackCmplxData(P6509_RCVR_PARAMS *rcvr_params,
                          register volatile int *inSamples,
                          register volatile int *outSamples,
                          register int num_samples)
{
    register int i;
    register int temp;
    register int temp1;

    /* Read out data from the fifo */

    for (i = 0; i < num_samples; i++)
    {
        if (rcvr_params->data_order == P65XX_IQ_ORDER)
        {
                temp = *inSamples++;
                *outSamples++ = temp >> 16;           /* Real(I) Value   */
                *outSamples++ = (temp << 16) >> 16;   /* Imag(Q) Value   */
        }
        else if (rcvr_params->data_order == P65XX_RECEIVER_ORDER)
        {
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

A.2 **init6509.c** (continued)

A.2.30 P6509UnpackCmplxData Function (continued)

```
            temp = *inSamples++;   /* Imag (Q) Value */
            temp1 = *inSamples++;    /* Real (I) Value */
            if (rcvr_params->rcvr_select == P65XX_ODD_RECEIVER)
            {
                  *outSamples++ = (temp1 << 16) >> 16;  /* Real(I) Value   */
                  *outSamples++ = (temp << 16) >> 16;   /* Imag(Q) Value   */
            }
            else
            {
                  *outSamples++ = temp1 >> 16;  /* Real(I) Value   */
                  *outSamples++ = temp >> 16;   /* Imag(Q) Value   */
            }
        }
    }
```

A.2.31 P6509ReadFifoReal Function

```
/*****************************************************************************
 Function: P6509ReadFifoReal
 Description:  This function reads real data from the fifo.
 Inputs:   rcvr_params -  Pointer to the P65XX_RCVR_PARAMS structure containing
                          the receiver settings
           p6509Reg - Pointer to the 6509's address table
           samples - Pointer to a data buffer to store data
           num_samples - Number of samples to be read
           inputFifo - Fifo to be read (1 - 8)
 Outputs:  Data is stored to the data buffer
 Returns:  samples - Address of the next available location in data buffer
 Notes:    Data is sign extended and right justified
*****************************************************************************/

int *P6509ReadFifoReal(P6509_RCVR_PARAMS *rcvr_params,
                       P6509_REG_ADDR *p6509Reg,
                       register volatile int *samples,
                       register int num_samples, unsigned int inputFifo)
{
    register volatile int *fifo = (int *)p6509Reg->Fifo +
                                  ((0x8000 * (inputFifo - 1)) / 4);
    int    num_samples_to_read;
    register int temp;
    register int i;
    while (num_samples > 0)
    {
        /* wait FIFO half-full */

           while ((*p6509Reg->Stat_Reg_2 & (0x100 << (inputFifo-1))) != 0);
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

A.2 **init6509.c** (continued)

A.2.31 P6509ReadFifoReal Function (continued)

```
            /* Determine how many samples left to read */

            if (num_samples <= P6509_FIFO_SIZE/2)
             num_samples_to_read = num_samples;
            else
                num_samples_to_read = P6509_FIFO_SIZE/2;

            /* Read out data from the fifo */

            for (i = 0; i < num_samples_to_read; i++)
            {
                temp = *fifo;               /* Store real data          */
                *samples++ = temp >> 16;    /* Real(I) Value            */
            }
            /* Decrement number of samples needed */
            num_samples -= num_samples_to_read;
    }
    return((int *)samples);
}
```

A.2.32 P6509ReadCommPortReal Function

```
/*************************************************************************
 Function: P6509ReadCommPortReal
 Description:  This function reads real data from the Comm Port.
 Inputs:   rcvr_params - Pointer to the P65XX_RCVR_PARAMS structure containing
                        the receiver settings
           samples - Pointer to a data buffer to store data
           num_samples - Number of samples to be read
           inputCommPort - Comm Port to be read (0 - 5)
 Outputs:  Data is stored to the data buffer in the following format:
              Unpacked Data -  Real,Real,Real,...        Sign Extended
                                                        Right Justified
              Packed, Time Ordered - Selected Rcvr Smpl1/Smpl2,Smpl1/Smpl2 ...
              Packed, Receiver Ordered - Real(Even)/Real(Odd),Real(Even),Real(Odd)..
 Returns:  samples - Address of the next available location in data buffer
 Notes:
*************************************************************************/

int *P6509ReadCommPortReal(P6509_RCVR_PARAMS *rcvr_params,
                           register volatile int *samples,
                           register int num_samples, unsigned int inputCommPort)
{
    volatile unsigned int *comStatus = (unsigned int *) 0x100040 +
                                                        0x10 * inputCommPort;
    register volatile int *comInput = (int *)comStatus + 1;
    int    num_samples_to_read;
    register int i;
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

A.2 **init6509.c** (continued)

A.2.32 P6509ReadCommPortReal Function (continued)

```
    if (rcvr_params->data_order == P65XX_TIME_ORDER)
        num_samples /= 2;

    /* Read out data from the Comm Port */

    for (i = 0; i < num_samples; i++)
    {
        while ((*comStatus & 0x1e00) == 0);
        if (rcvr_params->packing_mode == P65XX_UNPACKED)
        {
            *samples++ = *comInput >> 16;    /* Real Value    */
        }
        else if (rcvr_params->data_order == P65XX_RECEIVER_ORDER)
        {
               *samples++ = *comInput;     /* Real Even/Real Odd */
        }
        else if (rcvr_params->data_order == P65XX_TIME_ORDER)
        {
               *samples++ = *comInput;     /* Selected Rcvr Smpl2/Smpl1 */
        }
    }
    return((int *)samples);
}
```

A.2.33 P6509UnpackRealData Function

```
/*************************************************************************
 Function: P6509UnpackRealData
 Description: This function unpacks a real data buffer read from a Comm Port
              based on the packing mode selected.
 Inputs:   rcvr_params - Pointer to the P65XX_RCVR_PARAMS structure containing
                        the receiver settings
           inSamples - Pointer to a data buffer containing complex data
           outSamples - Pointer to the output data buffer
           num_samples - Number of samples to be read
 Outputs:  Data is stored to the output data buffer in the following format:
              Real,Real,Real,...
 Returns:
 Notes:    Data is sign extended and right justified
*************************************************************************/

void P6509UnpackRealData(P6509_RCVR_PARAMS *rcvr_params,
                         register volatile int *inSamples,
                         register volatile int *outSamples,
                         register int num_samples)
{
    register int i;
    register int temp;
    register int temp1;
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

A.2 **init6509.c** (continued)

A.2.33 **P6509UnpackRealData Function** (continued)

```
    if (rcvr_params->data_order == P65XX_TIME_ORDER)
        num_samples /= 2;

    /* Read out data from the fifo */

    for (i = 0; i < num_samples; i++)
    {
        if (rcvr_params->data_order == P65XX_TIME_ORDER)
        {
               temp = *inSamples++;
               *outSamples++ = (temp << 16) >> 16;   /* Real Sample 1   */
               *outSamples++ = temp >> 16;           /* Real Sample 2   */
        }
        else if (rcvr_params->data_order == P65XX_RECEIVER_ORDER)
        {
            temp = *inSamples++;     /* Real Value */
            if (rcvr_params->rcvr_select == P65XX_ODD_RECEIVER)
                   *outSamples++ = (temp << 16) >> 16;  /* Real(I) Value   */
            else
                   *outSamples++ = temp >> 16;          /* Real(I) Value   */
        }
    }
}
```

A.3 65xxcmd.h

This file defines constants used to communicate with the 65XX receiver using the communications ports or the VMEBus Interface.

```
#ifndef P65XX_CMD_H
#define P65XX_CMD_H

/* Command Word Format */

typedef union P65XX_CMD_WORD
  {
     struct
        {
           unsigned int  cmd                :12;
           unsigned int  handle             :8;
           unsigned int  wait_flag          :1;
           unsigned int  data_fmt           :2;
           unsigned int  spare              :9;
        } fields;
           unsigned long cmd_word;
           } P65XX_CMD_WORD;
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

A.3 **65xxcmd.h** (continued)

```
/* Command Word Macros */

#define P65XX_BUILD_CMD(cmd,handle) (cmd | (handle << 12) |\
                                    P65XX_NO_WAIT_COMPLETE | P65XX_DEFAULT_TYPE)
#define P65XX_BUILD_CMD_WAIT(cmd,handle) (cmd | (handle << 12) |\
                                    P65XX_WAIT_COMPLETE |\ P65XX_DEFAULT_TYPE)
#define P65XX_BUILD_CMD_IEEE(cmd,handle,wait_flag) (cmd | (handle << 12) |\
                                    wait_flag |\ P65XX_IEEE_FLOAT_TYPE)
#define P65XX_BUILD_CMD_INTEGER(cmd,handle,wait_flag) (cmd | (handle << 12) |\
                                    wait_flag |\ P65XX_INTEGER_TYPE)

/* Command Word Masks */

#define P65XX_CMD_MASK                0x00000FFFL
#define P65XX_HANDLE_MASK             0x000FF000L
#define P65XX_WAIT_FLAG_MASK          0x00100000L
#define P65XX_DATA_FMT_MASK           0x00600000L

/* Wait Flag Constants */

#define P65XX_NO_WAIT_COMPLETE        0x00000000L
#define P65XX_WAIT_COMPLETE           0x00100000L

/* Data Format Constants */

#define P65XX_DEFAULT_TYPE            0x00000000L
#define P65XX_IEEE_FLOAT_TYPE         0x00200000L
#define P65XX_INTEGER_TYPE            0x00400000L

/* Offset from the start of Global SRAM where commands are written from VME. This
   is a C40 Offset.  The actual VMEBus offset is P65XX_VME_COMMAND_ADDR * 4. */

#define P65XX_VME_COMMAND_ADDR        0x00003000L

/* Constant defining maximum number of Handles or Open Connections */

#define P65XX_NUM_DEV_HANDLES         16

/* Constant defining the maximum number of commands */

#define P65XX_NUM_COMMANDS            90

/* 65XX Commands  */

#define P65XX_OPEN_CONNECTION            1
#define P65XX_CLOSE_CONNECTION           2
#define P65XX_CLOSE_ALL_CONNECTIONS      3
#define P65XX_RESET_REGS                 4
#define P65XX_INIT_RCVRS                 5
#define P65XX_INIT_INPUT_CHANNELS        6
#define P65XX_SET_LO                     7
#define P65XX_SET_DECIMATION             8
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

A.3 65xxcmd.h (continued)

```
#define P65XX_SET_INP_CLK_RATE          9
#define P65XX_FLIP_SPECTRUM             10
#define P65XX_OFFSET_SPECTRUM           11
#define P65XX_SET_DATA_MODE             12
#define P65XX_SET_DATA_SRC              13
#define P65XX_SET_OUTPUT_SRC            14
#define P65XX_SET_PACKING_MODE          15
#define P65XX_SET_DATA_ORDER            16
#define P65XX_SELECT_RECEIVER           17
#define P65XX_TAG_ENABLE                18
#define P65XX_SET_THRESHOLD_DETECTION   19
#define P65XX_RESET_FIFO                20
#define P65XX_RELEASE_FIFO              21
#define P65XX_FLUSH_FIFO                22
#define P65XX_GET_RCVR_STATUS           23
#define P65XX_GET_RCVR_DATA             24
#define P65XX_STOP_RCVR_DATA            25
#define P65XX_MODIFY_INTR_MASK          26
#define P65XX_DISABLE_FIRMWARE          41
#define P65XX_LOOK                      42
#define P65XX_SET                       43
#define P65XX_GET_REVISION              44

/* 6504 Specific Commands */
/* 6505 Specific Commands */
/* 6508 Specific Commands */

#define P6508_SET_INPUT_RESET_MODE      71
#define P6508_SET_RCVR_WORD             72

/* 6509 Specific Commands */

#define P6509_SET_INPUT_SYNC_MODE       81
#define P6509_SET_RCVR_SYNC_MODE        82
#define P6509_SET_FILTER_RSP            83

/* 6510 Specific Commands */
/* 6514 Specific Commands */
/* 6516 Specific Commands */
/* The second parameter of each command is the input channel/receiver mask.
   The constants used to define the receivers or input channels are defined
   as:                                                                    */
/* Receiver Channels */

#define P65XX_RECEIVER_1                0x01
#define P65XX_RECEIVER_2                0x02
#define P65XX_RECEIVER_3                0x04
#define P65XX_RECEIVER_4                0x08
#define P65XX_RECEIVER_5                0x10
#define P65XX_RECEIVER_6                0x20
#define P65XX_RECEIVER_7                0x40
#define P65XX_RECEIVER_8                0x80
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

A.3 **65xxcmd.h** (continued)

```
/* Input Channels  */
/* Data Source - This Selection determines which input connector   */
/* contains the clock and data.                                      */

#define P65XX_INPUT_A                   0x01
#define P65XX_INPUT_B                   0x02
#define P65XX_INPUT_C                   0x04
#define P65XX_INPUT_D                   0x08

/* Data Modes */

#define P65XX_COMPLEX                   0
#define P65XX_REAL                      1

/* Packing Modes */

#define P65XX_PACKED                    0
#define P65XX_UNPACKED                  1

/* Data Order */

#define P65XX_RECEIVER_ORDER            0
#define P65XX_IQ_ORDER                  1       /* Complex Data Only */
#define P65XX_TIME_ORDER                1       /* Real Data Only    */

/* Channel Select */

#define P65XX_EVEN_RECEIVER             0       /* Receiver 2, 4, 6, 8 */
#define P65XX_ODD_RECEIVER              1       /* Receiver 1, 3, 5, 7 */

/* Flip Spectrum or Spectral Reversal */

#define P65XX_FLIP_OFF                  0
#define P65XX_FLIP_ON                   1

/* Offset Spectrum or Up Convert */

#define P65XX_OFFSET_OFF                0
#define P65XX_OFFSET_ON                 1

/* Data Transfer Modes */

#define P65XX_FIFO                      0
#define P65XX_COMM_PORT                 1

/* Tag Enable */

#define P65XX_TAG_OFF                   0
#define P65XX_TAG_ON                    1
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

A.3 **65xxcmd.h** (continued)

```
/* Wait on Ram Full - 6508/6510 Only */

#define P65XX_WAIT_RAM_FULL_OFF          0
#define P65XX_WAIT_RAM_FULL_ON           1

/* Threshold Level */

#define P65XX_THRSHLD_DETECT_OFF         10x00
#define P65XX_ONE_HALF_FSCALE            10x80
#define P65XX_FIVE_EIGHTS_FSCALE         10xA0
#define P65XX_THREE_FOURTHS_FSCALE       10xC0
#define P65XX_SEVEN_EIGHTS_FSCALE        10xE0

/* Threshold Crossing Count */

#define P65XX_CROSS_CNT_2                0
#define P65XX_CROSS_CNT_8                1
#define P65XX_CROSS_CNT_32               2
#define P65XX_CROSS_CNT_128              3
#define P65XX_CROSS_CNT_512              4
#define P65XX_CROSS_CNT_2048             5
#define P65XX_CROSS_CNT_8192             6
#define P65XX_CROSS_CNT_32768            7

/* Interrupt Command - Parameter 1 */

#define P65XX_DISABLE_INTERRUPT          0x0
#define P65XX_GEN_CP_2_OUTPUT            0x2
#define P65XX_GEN_CP_3_OUTPUT            0x3
#define P65XX_GEN_CP_4_OUTPUT            0x4
#define P65XX_GEN_CP_5_OUTPUT            0x5
#define P65XX_GEN_VME_INTR               0x10
#define P65XX_EDGE_TRIG_INTR             0x1000
#define P65XX_LEVEL_TRIG_INTR            0x0000
#define P65XX_ENABLE_INTERRUPT           0x4000
#define P65XX_MASK_ON_OCCURRENCE         0x8000

/* Interrupt Command Conditions - Parameter 2 */

#define P65XX_FIFO_NOT_EMPTY_1           0x1
#define P65XX_FIFO_NOT_EMPTY_2           0x2
#define P65XX_FIFO_NOT_EMPTY_3           0x4
#define P65XX_FIFO_NOT_EMPTY_4           0x8
#define P65XX_FIFO_NOT_EMPTY_5           0x10
#define P65XX_FIFO_NOT_EMPTY_6           0x20
#define P65XX_FIFO_NOT_EMPTY_7           0x40
#define P65XX_FIFO_NOT_EMPTY_8           0x80
#define P65XX_FIFO_HALF_FULL_1           0x100
#define P65XX_FIFO_HALF_FULL_2           0x200
#define P65XX_FIFO_HALF_FULL_3           0x400
#define P65XX_FIFO_HALF_FULL_4           0x800
#define P65XX_FIFO_HALF_FULL_5           0x1000
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

A.3 65xxcmd.h (continued)

```
#define P65XX_FIFO_HALF_FULL_6          0x2000
#define P65XX_FIFO_HALF_FULL_7          0x4000
#define P65XX_FIFO_HALF_FULL_8          0x8000
#define P65XX_FIFO_FULL_1               0x10000
#define P65XX_FIFO_FULL_2               0x20000
#define P65XX_FIFO_FULL_3               0x40000
#define P65XX_FIFO_FULL_4               0x80000
#define P65XX_FIFO_FULL_5               0x100000
#define P65XX_FIFO_FULL_6               0x200000
#define P65XX_FIFO_FULL_7               0x400000
#define P65XX_FIFO_FULL_8               0x800000
#define P65XX_CLOCK_NOT_READY_A         0x1000000
#define P65XX_CLOCK_NOT_READY_B         0x2000000
#define P65XX_CLOCK_NOT_READY_C         0x4000000
#define P65XX_CLOCK_NOT_READY_D         0x8000000
#define P65XX_THRSHLD_DETECT_A          0x10000000
#define P65XX_THRSHLD_DETECT_B          0x20000000
#define P65XX_THRSHLD_DETECT_C          0x40000000
#define P65XX_THRSHLD_DETECT_D          0x80000000

/* Receiver Data Type and Latency */

#define P65XX_BUFFERED_DATA             0x1
#define P65XX_NON_BUFFERED_DATA         0x2
#define P65XX_COMPLEX_DATA              0x0
#define P65XX_REAL_DATA                 0x100

/* 6508 Specific Constants */
/* Input Channel Sync Modes */

#define P6508_SW_RESET_ON               0x0
#define P6508_SW_RESET_OFF              0x2
#define P6508_FP_RESET_IN_SRC           0x0
#define P6508_SW_RESET_IN_SRC           0x4
#define P6508_RESET_OUT_DIS             0x0
#define P6508_RESET_OUT_ENA             0x8

/* 6509 Specific Constants */
/* Input Channel Sync Modes */

#define P6509_SW_AS_ON                  0x0
#define P6509_SW_AS_OFF                 0x1
#define P6509_SW_SS_ON                  0x0
#define P6509_SW_SS_OFF                 0x2
#define P6509_FP_SYNC_SRC               0x0
#define P6509_SW_SYNC_SRC               0x4
#define P6509_SYNC_OUT_DIS              0x0
#define P6509_SYNC_OUT_ENA              0x8
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

A.3 **65xxcmd.h** (continued)

```
/* Receiver Channel Sync Modes - The description for these bits are defined
   in the GC1011A Data Sheet */

#define P6509_GC_SS_OFF                 0x01
#define P6509_GC_AS_ON                  0x02
#define P6509_GC_AS_MUX                 0x04
#define P6509_GC_LD_FREQ                0x08
#define P6509_GC_AS_FREQ                0x10
#define P6509_GC_SS_DIAG                0x20
#define P6509_GC_SYNC_TEST              0x40
#define P6509_GC_SS_MUX                 0x80

/* Filter Selection */

#define P6509_80_PERCENT                0x00
#define P6509_90_PERCENT                0x01

/* 65XX Command Status */

#define P65XX_COMMAND_COMPLETE          0
#define P65XX_COMMAND_ERROR             1
#define P65XX_PARAM_ERROR               2
#define P65XX_OPEN_ERROR                3

/* 65XX Command Function Prototypes */

unsigned long P65XXSendCommand(unsigned int comm_port, unsigned int command,
                               unsigned int num_params, void *cmdParams,
                               unsigned int input_rcvr_mask);
unsigned long P65XXSendCommandWait(unsigned int comm_port, unsigned int command,
                                   unsigned int num_params, void *cmdParams,
                                   unsigned int input_rcvr_mask,
                                   unsigned long *numRData,
                                   unsigned long *returnData);
void P65XXReadCommPortData(unsigned int data_port,
                           register volatile int *insamples,
                           register unsigned long num_samples);
void P65XXUnpackCmplxData(register int *inSamples, register int *outSamples,
                           register unsigned long num_samples,
                           register unsigned int rcvr_index,
                           register unsigned int numRcvrs);
void P65XXUnpackRealData(register int *inSamples, register int *outSamples,
                         register unsigned long num_samples,
                         register unsigned int rcvr_index,
                         register unsigned int numRcvrs);
#endif
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

A.4 p65xxcmd.c - Comm Port Interface

```
/*************************************************************************
*
*   File : p65xxcmd.c
*
*   Description: The file contains routines for sending commands to the 65XX
*                receiver boards via the Comm Port Interface.  Routines for
*                reading and unpacking Comm Port data provided by the
*                P65XX_GET_RCVR_DATA command hav also been included.
*
*   $Revision:$   $Date:$
*
*   $History:$
*
*************************************************************************/

#include "65xxcmd.h"
```

A.4.1 P65XXSendCommand Function

```
/*************************************************************************
 Function: P65XXSendCommand
 Description:  This routine generates a command to the 65XX Board via the
               specified comm port.
 Inputs:  comm_port - Communication Port used for sending commands
          command   - Command being sent
          num_params - The number of parameters which must follow command
          cmdParams - Pointer to the command parameters
          input_rcvr_mask - Input Channel or Receiver mask
 Outputs:
 Returns: Command Complete status
 Notes: Returns immediately to caller without waiting for a reply
*************************************************************************/

unsigned long P65XXSendCommand(unsigned int comm_port, unsigned int command,
                               unsigned int num_params, void *cmdParams,
                               unsigned int input_rcvr_mask)
{
  unsigned int   i;
  volatile unsigned long *com = (volatile unsigned long *)0x100040 + 0x10 * comm_port;
  *(com + 2) = 0x12345678;
  *(com + 2) = command;
  *(com + 2) = input_rcvr_mask;
  for (i = 0; i < num_params; ++i)
  *(com + 2) = *((unsigned long *)cmdParams + i);
  return(P65XX_COMMAND_COMPLETE);
}
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

A.4 p65xxcmd.c (continued)

A.4.2 P65XXSendCommandWait Function

```
/*****************************************************************************
 Function: P65XXSendCommandWait
 Description:  This routine generates a command to the 65XX Board via the
               specified comm port and waits for a response.
 Inputs:  comm_port - Communication Port used for sending commands
          command   - Command being sent
          num_params - The number of parameters which must follow command
          cmdParams - Pointer to the command parameters
          input_rcvr_mask - Input Channel or Receiver mask
          numRdata - Pointer to store the number of returned data
          returnData - Buffer pointer to store the return data
 Outputs: numRdata - Number of returnned data
          returnData - Returned Data
 Returns: Status - Status of the Command
 Notes:
*****************************************************************************/

unsigned long P65XXSendCommandWait(unsigned int comm_port, unsigned int command,
                                   unsigned int num_params, void *cmdParams,
                                   unsigned int input_rcvr_mask,
                                   unsigned long *numRData,
                                   unsigned long *returnData)
{
  unsigned int i;
  unsigned int status;
  volatile unsigned long *com = (volatile unsigned long *)0x100040 + 0x10 * comm_port;
  *(com + 2) = 0x12345678;
  *(com + 2) = command;
  *(com + 2) = input_rcvr_mask;
  for (i = 0; i < num_params; ++i)
  *(com + 2) = *((unsigned long *)cmdParams + i);

  /* Wait for Input Data */

  while ((*com & 0x1e00) == 0);

  /* Read Status and return data */

  status = *(com + 1);
  if ((numRData != NULL) && (returnData != NULL))
     {
        *numRData = status >> 16;
        for (i = 0; i < *numRData; ++i)
        *returnData++ = *(com + 1);
     }
     return(status & 0xffff);
}
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

A.4 **p65xxcmd.c** (continued)

A.4.3 P65XXReadCommPortData Function

```
/*****************************************************************************
 Function: P65XXReadCommPortData
 Description:  This function reads the specified number of samples from the
               comm port.
 Inputs:  data_port - Comm Port to be read (0 - 5)
          insamples - Buffer pointer to store data
          num_samples - Number of samples to be read
 Outputs: data samples - Format depends on the data mode selected in the
                         P65XX_GET_RCVR_DATA Command
 Returns:
 Notes:
*****************************************************************************/

void P65XXReadCommPortData(unsigned int data_port,
                           register volatile int *insamples,
                           register unsigned long num_samples)
{
  register volatile unsigned int *com_status = (unsigned int *)0x100040 +
                                                        0x10 * data_port;
  register volatile unsigned int *com = com_status + 1;
  register unsigned long i;

/* Read in the data */

  for (i = 0; i < num_samples; ++i)
     {

       /* Wait for the Input Comm Port to have some data */

       while ((*com_status & 0x1e00) == 0);
       *insamples++ = *com;
     }
}
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

A.4 **p65xxcmd.c** (continued)

A.4.4 P65XXUnpackCmplxData Function

```
/*****************************************************************************
 Function: P65XXUnpackCmplxData
 Description:  This function unpacks a complex data buffer containing
               receiver data from N receivers.
 Inputs:  inSamples - Pointer to a data buffer containing complex data
          outSamples - Pointer to the output data buffer
          num_samples - Number of samples to be read
          rcvr_index - first sample index into input sample buffer
          numRcvrs - The number of receivers whose data is stored in the input
                     buffer
 Outputs:    Data is stored to the data buffer in the following format:
               Real,Imag,Real,Imag......
 Returns:
 Notes:   Data is sign extended and right justified
*****************************************************************************/

void P65XXUnpackCmplxData(register int *inSamples, register int *outSamples,
                          register unsigned long num_samples,
                          register unsigned int rcvr_index,
                          register unsigned int numRcvrs)
{
  register unsigned long i;
  inSamples += rcvr_index;
  for (i = 0; i < num_samples; ++i, inSamples += numRcvrs)
  {
     *outSamples++ = *inSamples >> 16;                /* Real(I) */
     *outSamples++ = (*inSamples << 16) >> 16;        /* Imag(Q) */
  }
}
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

A.4 **p65xxcmd.c** (continued)

A.4.5 P65XXUnpackRealData

```
/*****************************************************************************
 Function: P65XXUnpackRealData
 Description:  This function unpacks a real data buffer containing
               receiver data from N receivers.
 Inputs:  inSamples - Pointer to a data buffer containing real data
          outSamples - Pointer to the output data buffer
          num_samples - Number of samples to be read
          rcvr_index - first sample index into input sample buffer
          numRcvrs - The number of receivers whose data is stored in the input
                     buffer
 Outputs: Data is stored to the data buffer in the following format:
             Real,Real,Real,...
 Returns:
 Notes: Data is sign extended and right justified
*****************************************************************************/

void P65XXUnpackRealData(register int *inSamples, register int *outSamples,
                         register unsigned long num_samples,
                         register unsigned int rcvr_index,
                         register unsigned int numRcvrs)
{
  register unsigned long i;
  inSamples += rcvr_index;
  for (i = 0; i < num_samples; ++i, inSamples += numRcvrs)
  {
     *outSamples++ = *inSamples;                /* Real(I) */
  }
}
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

A.5 main.c

```
/*************************************************************************
*
*   File : main.c
*
*   Description: This program demonstrates how to initialize and receive data
*                using the Comm Port Interface for the 6509 or 6508 Digital
*                Receiver Boards.
*
*   $Revision:$   $Date:$
*
*   $History:$
*
*************************************************************************/

#include <stdlib.h>
#include <snstdio.h>
#include "math.h"
#include "65xxcmd.h"

/* Define the Digital Receiver Board Model */

#define P6508     0
#define P6509     1
#if P6509
#include "6509.h"
#define P65XX_NUM_RCVRS         P6509_NUM_RCVRS
#define P65XX_NUM_INPUTS        P6509_NUM_INPUT_CHANNELS
typedef P6509_RCVR_PARAMS       P65XX_RCVR_PARAMS;
typedef P6509_INPUT_PARAMS      P65XX_INPUT_PARAMS;
#define RCVR_PARAM_TBL_SIZE     sizeof(P6509_RCVR_PARAMS)
#define INPUT_PARAM_TBL_SIZE    sizeof(P6509_INPUT_PARAMS)
#define ReadCommPortCmplx       P6509ReadCommPortCmplx
#define ReadCommPortReal        P6509ReadCommPortReal
#define UnpackCmplxData         P6509UnpackCmplxData
#define UnpackRealData          P6509UnpackRealData
#endif
#if P6508
#include "6508.h"
#define P65XX_NUM_RCVRS         P6508_NUM_RCVRS
#define P65XX_NUM_INPUTS        P6508_NUM_INPUT_CHANNELS
typedef P6508_RCVR_PARAMS       P65XX_RCVR_PARAMS;
typedef P6508_INPUT_PARAMS      P65XX_INPUT_PARAMS;
#define RCVR_PARAM_TBL_SIZE     sizeof(P6508_RCVR_PARAMS)
#define INPUT_PARAM_TBL_SIZE    sizeof(P6508_INPUT_PARAMS)
#define ReadCommPortCmplx       P6508ReadCommPortCmplx
#define ReadCommPortReal        P6508ReadCommPortReal
#define UnpackCmplxData         P6508UnpackCmplxData
#define UnpackRealData          P6508UnpackRealData
#endif
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

A.5 **main.c** (continued)

```
/* The following constants define some of the parameters to this program */
/* OUTPUT_SRC - Defines whether data is automatically sent over the receivers
          Output Comm Ports or read from the Fifo via the VMEBus or the Local C40.
          When the Output Src is set to P65XX_FIFO, this example will request to
          the 65XX Reciever board to sent the specified receivers data to the
          front panel Input Comm Port. This allows all receiver data to be output
          to a single comm port.                                                 */

#define OUTPUT_SRC            P65XX_COMM_PORT

/* RCVR_MASK - Defines which receivers are initialized and supplying data.
          When the OUTPUT_SRC is P65XX_COMM_PORTS, only a single receiver pair
          can be specified(i.e. 1 & 2, 3 & 4, 5 & 6, 7 & 8). When the OUTPUT_SRC
          is P65XX_FIFO, any combination of receivers can be specified.    */

#define RCVR_MASK             (P65XX_RECEIVER_3 | P65XX_RECEIVER_4)

/* DATA_SOURCE - Defines which input channel is suppliying data to the receiver */

#define DATA_SOURCE           P65XX_INPUT_A

/* PACKING_MODE - Defines whether Packed or Unpacked Data is sent to the front
                  panel Output Comm Ports.  This is only applicable when
                  OUTPUT_SRC is P65XX_COMM_PORT.    */

#define PACKING_MODE          P65XX_UNPACKED

/* DATA_MODE - Defines whether the data supplied by the receiver is real or
               complex.                                */

#define DATA_MODE          P65XX_COMPLEX

/* DECIMATION_RATE - Defines the output data rate of the receiver.
                     Output Rate = (INPUT_CLOCK_RATE/ DECIMATION_RATE) */

#define DECIMATION_RATE    128

/* LO - Defines the tuning frequency of the receivers */

#define LO                 (float) 12.0e6

/* INPUT_CLOCK_RATE - Defines the input clock rate to the receivers */

#define INPUT_CLOCK_RATE   (float) 40.0e6

/* CMD_COMM_PORT - Defines the C40 Comm Port which will be used to send commands
                   to the Digital Receiver Board. */

#define CMD_COMM_PORT      2
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

A.5 main.c (continued)

```
/* DATA_COMM_PORT - Defines the C40 Comm Port which will receive data from the
                    Digital Receiver Board. */

#define DATA_COMM_PORT      0

/* Number of Samples to acquire - In complex Mode, this program refers to */
/* one sample as one real plus one imaginary sample.                      */

#define NUM_CMPLX_SAMPLES       512
#define NUM_REAL_SAMPLES        1024

/* Data Buffers for storing received data */

int     inSamples[NUM_REAL_SAMPLES * P65XX_NUM_RCVRS];
int     Samples[NUM_REAL_SAMPLES * 2];
```

A.5.1 set65XXRcvrParams Function

```
/*****************************************************************************
 Function: set65XXRcvrParams
 Description:   This routine initializes the Receiver Parameter Table for a
                specified receiver.
 Inputs:  params - Pointer to a P65XX_RCVR_PARAMS table entry.
          rcvr - Receiver being setup(1 - 8)
 Outputs:
 Returns:
 Notes:
*****************************************************************************/

void set65XXRcvrParams(P65XX_RCVR_PARAMS *params, int rcvr)
{
  params->decimation_rate = DECIMATION_RATE;
  params->data_mode = DATA_MODE;
  params->data_source = DATA_SOURCE;
  params->lo = LO;
  params->input_clock_rate = INPUT_CLOCK_RATE;
  params->packing_mode = PACKING_MODE;
  params->data_transfer = OUTPUT_SRC;
  params->data_order = P65XX_RECEIVER_ORDER;
  params->rcvr_select = rcvr & 0x1;
  params->tag_enable = P65XX_TAG_OFF;
  params->flip_spectrum = P65XX_FLIP_OFF;
  params->offset_spectrum = P65XX_OFFSET_ON;
#if P6509
  params->filter_select = P6509_80_PERCENT;
  params->rcvr_sync_mode = P6509_GC_SS_OFF | P6509_GC_LD_FREQ;
#endif
#if P6508
  params->wait_ram_full = P65XX_WAIT_RAM_FULL_ON;
#endif
}
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

A.5 **main.c** (continued)

A.5.2 set65XXInputParams

```
/*****************************************************************************
 Function: set65XXInputParams
 Description:  This routine initializes the Input Channel Parameter Table for
               a specified input channel.
 Inputs:  params - Pointer to a P65XX_INPUT_PARAMS table entry.
          input - Input Channel being setup
 Outputs:
 Returns:
 Notes:
*****************************************************************************/

void set65XXInputParams(P65XX_INPUT_PARAMS *params, int input)
{
    params->threshold_level = P65XX_THRSHLD_DETECT_OFF;
    params->threshold_cross_cnt = P65XX_CROSS_CNT_2;
#if P6508
    params->input_reset_mode = P6508_SW_RESET_IN_SRC | P6508_SW_RESET_OFF |
                               P6508_RESET_OUT_DIS;
#endif
#if P6509
    params->input_sync_mode = P6509_SW_SYNC_SRC | P6509_SW_AS_OFF |
                              P6509_SW_SS_OFF;
#endif
}
```

A.5.3 main function

```
/*****************************************************************************
 Function: main
 Description: This program initializes the specified receivers and
              acquires data.
*****************************************************************************/

main()
{
    register int          i;
    register int          j;
    int                   temp;
    unsigned int          cmdParams[2];
    unsigned long         status;
    int                   *samples;
    P65XX_RCVR_PARAMS     p65xxRcvrParams[P65XX_NUM_RCVRS];
    P65XX_INPUT_PARAMS    p65xxInputParams[P65XX_NUM_INPUTS];
    unsigned int          handle[P65XX_NUM_RCVRS] = {0, 0, 0, 0, 0, 0, 0, 0};
    unsigned long         returnData[2];
    unsigned long         numRData;
    volatile unsigned int *com;
    unsigned long         a_rcvr;
    unsigned long         numRcvrs = 0;
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

A.5 **main.c** (continued)

A.5.3 **main function** (continued)

```
unsigned long         rcvrIndex;

/* Initialize SNIO interface */

if (snstdio_init(NULL) != OK)
    exit(1);
atexit((void (*)())snstdio_shutdown);

/* Clear out Command Comm Port */

com = (volatile unsigned int *)(0x100040 + 0x10 * CMD_COMM_PORT);
while ((*com & 0x1e00) != 0)
    temp = *(com + 1);

/* Close all prior open connections */

if ((status = P65XXSendCommandWait(CMD_COMM_PORT,
                        P65XX_BUILD_CMD_WAIT(P65XX_CLOSE_ALL_CONNECTIONS, 0),
                        0, NULL, 0, &numRData, returnData)) !=
                                              P65XX_COMMAND_COMPLETE)
{
    printf("Error Closing All Connections - Status = %lx\n", status);
    exit(1);
}

/* Reset all 65XX Registers */

if ((status = P65XXSendCommandWait(CMD_COMM_PORT,
                                    P65XX_BUILD_CMD_WAIT(P65XX_RESET_REGS, 0),
                                    0, NULL, 0, &numRData, returnData)) !=
                                             P65XX_COMMAND_COMPLETE)
{
    printf("Error Resetting Registers - Status = %lx\n", status);
    exit(1);
}

/* Initialize Input Channels */

for (i = 0; i < P65XX_NUM_INPUTS; ++i)
{
     /* Initialize Input Channel Parameter Data structure */
     set65XXInputParams(&p65xxInputParams[i], i);
     /* Initialize the Input Channel */
     if ((status = P65XXSendCommandWait(CMD_COMM_PORT,
                              P65XX_BUILD_CMD_WAIT(P65XX_INIT_INPUT_CHANNELS, 0),
                              sizeof(P65XX_INPUT_PARAMS),
                              (void *)&p65xxInputParams[i], 1 << i,
                               &numRData, returnData)) != P65XX_COMMAND_COMPLETE)
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

A.5 **main.c** (continued)

A.5.3 **main function** (continued)

```
{
            printf("Error Initializing Input %c - Status = %lx\n", 'A' + i,
                   status);
            exit(1);
        }
    }

    /* Open a connection and initialize each receiver  */

    for (i = 0; i < P65XX_NUM_RCVRS; ++i)
    {
        if (RCVR_MASK & (1 << i))
        {
            /* Initialize Parameter Data structure */

            set65XXRcvrParams(&p65xxRcvrParams[i], i + 1);

            /* Establish Connection for this receiver */

            if ((status = P65XXSendCommandWait(CMD_COMM_PORT,
                                P65XX_BUILD_CMD_WAIT(P65XX_OPEN_CONNECTION, 0),
                                0, NULL, 1 << i, &numRData, returnData)) !=
                                                P65XX_COMMAND_COMPLETE)
            {
                printf("Error Opening Connection Rcvr %d - Status = %lx\n",
                               i + 1, status);
                exit(1);
            }

            /* Save the handle corresponding to this connection */

            handle[i] = *returnData;

            /* Initialize the Receiver */

            if ((status = P65XXSendCommandWait(CMD_COMM_PORT,
                                P65XX_BUILD_CMD_WAIT(P65XX_INIT_RCVRS, handle[i]),
                                sizeof(P65XX_RCVR_PARAMS),
                                (void *)&p65xxRcvrParams[i], 1 << i,
                                &numRData, returnData)) != P65XX_COMMAND_COMPLETE)
            {
                printf("Error Initializing Rcvr %d - Status = %lx\n",
                               i + 1, status);
                exit(1);
            }

            /* Save an index into one of the valid receivers data structures */

            a_rcvr = i;
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

A.5 **main.c** (continued)

A.5.3 **main function** (continued)

```
            /* Increment the number of receivers present in RCVR_MASK */

            ++numRcvrs;
        }
    }

    /* Main Data Acquisition and Processing Loop */

    while (1)
    {
        /* Reset all the Receiver Fifos */

        if ((status = P65XXSendCommandWait(CMD_COMM_PORT,
                            P65XX_BUILD_CMD_WAIT(P65XX_RESET_FIFO, handle[a_rcvr]),
                            0, NULL, RCVR_MASK, &numRData, returnData)) !=
                                                    P65XX_COMMAND_COMPLETE)
        {
            printf("Error Resetting Fifos %d - Status = %lx\n", i, status);
            exit(1);
        }

        /* Clear out old data from the Data Comm Port */

        com = (volatile unsigned int *)(0x100040 + 0x10 * DATA_COMM_PORT);
        while ((*com & 0x1e00) != 0)
            temp = *(com + 1);
#if OUTPUT_SRC == P65XX_FIFO

        /* When the OUTPUT_SRC is P65XX_FIFO, set up the 65XX board to output
           the receiver data to a specified Comm Port on the 65XX Board.
           The mapping of the front panel input Comm Ports to C40 Comm Ports is:
               1 & 2 IN - C40 Comm Port 2
               3 & 4 IN - C40 Comm Port 3
               5 & 6 IN - C40 Comm Port 4
               7 & 8 IN - C40 Comm Port 5                                    */

        cmdParams[0] = 5;     /* Comm Port 5 */

        /* Output the specified data tye(REAL, COMPLEX) as soon as it is
           available.                                                        */

        cmdParams[1] = (p65xxRcvrParams[a_rcvr].data_mode << 8) |
                        P65XX_NON_BUFFERED_DATA;
        if ((status = P65XXSendCommandWait(CMD_COMM_PORT,
                                    P65XX_BUILD_CMD_WAIT(P65XX_GET_RCVR_DATA, 0),
                                    2, cmdParams, RCVR_MASK, &numRData,
                                    returnData)) != P65XX_COMMAND_COMPLETE)
        {
                printf("Error Getting Rcvr Data - Status = %lx\n", status);
                exit(1);
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

A.5 **main.c** (continued)

A.5.3 **main function** (continued)

```
        }
#endif

    /* Release the Fifos and begin acquisition */

        P65XXSendCommand(CMD_COMM_PORT,
                         P65XX_BUILD_CMD(P65XX_RELEASE_FIFO, handle[a_rcvr]),
                         0, NULL, RCVR_MASK);
#if OUTPUT_SRC == P65XX_FIFO

        /* Read the data for all receivers from a single Comm Port */

        if (p65xxRcvrParams[a_rcvr].data_mode == P65XX_COMPLEX)
            P65XXReadCommPortData(DATA_COMM_PORT, inSamples,
                                  NUM_CMPLX_SAMPLES * numRcvrs);
        else
            P65XXReadCommPortData(DATA_COMM_PORT, inSamples,
                                  NUM_REAL_SAMPLES * numRcvrs);

        /* After all the necessary data is acquired, turn it OFF */

        if ((status = P65XXSendCommandWait(CMD_COMM_PORT,
                                    P65XX_BUILD_CMD_WAIT(P65XX_STOP_RCVR_DATA, 0),
                                    0, NULL, RCVR_MASK, &numRData, returnData)) !=
                                                P65XX_COMMAND_COMPLETE)
        {
            printf("Error Stopping Rcvr Data - Status = %lx\n", status);
            exit(1);
        }

        /* Initialize receiver index */

        rcvrIndex = 0;
#endif
#if OUTPUT_SRC == P65XX_COMM_PORT

    /* Read the data from the 65XX Fifo or Comm Port */

        if (p65xxRcvrParams[a_rcvr].data_mode == P65XX_COMPLEX)
            (void) ReadCommPortCmplx(&p65xxRcvrParams[a_rcvr], inSamples,
                                     NUM_CMPLX_SAMPLES, DATA_COMM_PORT);
        else
            (void) ReadCommPortReal(&p65xxRcvrParams[a_rcvr], inSamples,
                                    NUM_REAL_SAMPLES, DATA_COMM_PORT);
#endif
        /* For Each Receiver Pair, Unpack and Process the data  */
        for (i = 0; i < P65XX_NUM_RCVRS; i+=2)
        {
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

A.5 **main.c** (continued)

A.5.3 **main function** (continued)

```
            /* For Each Receiver in Pair */

            for (j = 0; j < 2; ++j)
            {
                if (RCVR_MASK & (1 << (i + j)))
                {
#if OUTPUT_SRC == P65XX_COMM_PORT
                    if (p65xxRcvrParams[i + j].packing_mode == P65XX_PACKED)
                    {
                        if (p65xxRcvrParams[i + j].data_mode == P65XX_COMPLEX)
                            UnpackCmplxData(&p65xxRcvrParams[i + j], inSamples,
                                            Samples, NUM_CMPLX_SAMPLES);
                        else
                            UnpackRealData(&p65xxRcvrParams[i + j],
                                           inSamples, Samples, NUM_REAL_SAMPLES);
                        samples = Samples;
                    }
                    else
                        samples = inSamples;
#endif
#if OUTPUT_SRC == P65XX_FIFO

                    /* Unpack the data stored in the data buffer */

                    if (p65xxRcvrParams[i + j].data_mode == P65XX_COMPLEX)
                        P65XXUnpackCmplxData(inSamples, Samples, NUM_CMPLX_SAMPLES,
                                             rcvrIndex, numRcvrs);
                    else
                        P65XXUnpackRealData(inSamples, Samples, NUM_REAL_SAMPLES,
                                            rcvrIndex, numRcvrs);
                    samples = Samples;

                    /* Increment receiver index */

                    ++rcvrIndex;
#endif

                    /* Add processing for receiver channel here */

                }
            }
        }
    }
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

A.5 **main.c** (continued)

A.5.3 **main function** (continued)

```
    /* Close the connection for each open connection */

    for (i = 0; i < P65XX_NUM_RCVRS; ++i)
    {
        if (RCVR_MASK & (1 << i))
        {

            /* Establish Connection for this receiver */

            if ((status = P65XXSendCommandWait(CMD_COMM_PORT,
                          P65XX_BUILD_CMD_WAIT(P65XX_CLOSE_CONNECTION, handle[i]),
                          0, NULL, 0, &numRData, returnData)) !=
                                                  P65XX_COMMAND_COMPLETE)
            {
                printf("Error Closing Connection Rcvr %d - Status = %lx\n",
                               i + 1, status);
                exit(1);
            }
        }
    }
}
```

A.6 main65XX.c - VMEBus only

```
/************************************************************************
*
*   File : main65XX.c
*
*   Description: This program demonstrates how to initialize and receive data
*                using the VMEBus Interface for the 6509 or 6508 Digital
*                Receiver Boards.
*
*   $Revision:$   $Date:$
*
*   $History:$
*
************************************************************************/

#include "vxWorks.h"
#include "stdio.h"
#include "stdlib.h"
#include "sysLib.h"
#include "msgQLib.h"
#include "iv.h"
#include "vme.h"
#include "taskLib.h"
#include "usrLib.h"
#include "cmdvxw.h"
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

A.6 main65XX.c - VMEBus only (continued)

```
#include "65xxcmd.h"

/* Define the Digital Receiver Board Model */

#define P6508     0
#define P6509     1
#if P6509
#include "6509.h"
#define P65XX_NUM_RCVRS         P6509_NUM_RCVRS
#define P65XX_NUM_INPUTS        P6509_NUM_INPUT_CHANNELS
typedef P6509_REG_ADDR          P65XX_REG_ADDR;
typedef P6509_RCVR_PARAMS       P65XX_RCVR_PARAMS;
typedef P6509_INPUT_PARAMS      P65XX_INPUT_PARAMS;
#define RCVR_PARAM_TBL_SIZE     sizeof(P6509_RCVR_PARAMS)
#define INPUT_PARAM_TBL_SIZE    sizeof(P6509_INPUT_PARAMS)
#define InitRegAddr             P6509InitRegAddr
#endif
#if P6508
#include "6508.h"
#define P65XX_NUM_RCVRS         P6508_NUM_RCVRS
#define P65XX_NUM_INPUTS        P6508_NUM_INPUT_CHANNELS
typedef P6508_REG_ADDR          P65XX_REG_ADDR;
typedef P6508_RCVR_PARAMS       P65XX_RCVR_PARAMS;
typedef P6508_INPUT_PARAMS      P65XX_INPUT_PARAMS;
#define RCVR_PARAM_TBL_SIZE     sizeof(P6508_RCVR_PARAMS)
#define INPUT_PARAM_TBL_SIZE    sizeof(P6508_INPUT_PARAMS)
#define InitRegAddr             P6508InitRegAddr
#endif

/* The following constants define some of the parameters to this program */
/* RCVR_MASK - Defines which receivers are initialized and supplying data.  */

#define RCVR_MASK      (P65XX_RECEIVER_1 | P65XX_RECEIVER_2 | P65XX_RECEIVER_3 |\
                        P65XX_RECEIVER_4 | P65XX_RECEIVER_5 | P65XX_RECEIVER_6 |\
                        P65XX_RECEIVER_7 | P65XX_RECEIVER_8)

/* DATA_SOURCE - Defines which input channel is suppliying data to the receiver */

#define DATA_SOURCE          P65XX_INPUT_A

/* DATA_MODE - Defines whether the data supplied by the receiver is real or
               complex.                                                    */

#define DATA_MODE            P65XX_COMPLEX

/* DECIMATION_RATE - Defines the output data rate of the receiver.
                     Output Rate = (INPUT_CLOCK_RATE/ DECIMATION_RATE) */

#define DECIMATION_RATE      128

/* LO - Defines the tuning frequency of the receivers                  */
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

A.6 main65XX.c - VMEBus only (continued)

```
#define LO                  12000000L

/* INPUT_CLOCK_RATE - Defines the input clock rate to the receivers     */

#define INPUT_CLOCK_RATE    150000000L

/* Modes of Operation */

#define POLLING   0
#define INTERRUPT 1

/* A16 and A32 Base Addresses */

#define P65XX_A16_BASE_ADDR    0xff00
#define P65XX_VME_BASE_ADDR    0x80000000L
#define P65XX_VME_ACCESS_MODE  VME_AM_EXT_USR_DATA

/* Interrupt Parameters */

#define P65XX_INTR_LEVEL       3
#define P65XX_INTR_VECTOR      208

/* Number of Samples to acquire - In complex Mode, this program refers to */
/* one sample as one real plus one imaginary sample.                       */

#define NUM_CMPLX_SAMPLES      512
#define NUM_REAL_SAMPLES       1024

/* Data Buffers for storing received data */

int     inSamples[NUM_REAL_SAMPLES * P65XX_NUM_RCVRS];
int     Samples[NUM_REAL_SAMPLES * 2];
volatile char    FifoFullIntr;
void fifoFullIntr();
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

A.6 main65XX.c - VMEBus only (continued)

A.6.1 set65XXRcvrParams Function- VMEBus only

```
/*****************************************************************************
 Function: set65XXRcvrParams

 Description:  This routine initializes the Receiver Parameter Table for
               a specified receiver.
 Inputs:  params - Pointer to a P65XX_RCVR_PARAMS table entry.
          rcvr - Receiver being setup(1 - 8)
 Outputs:
 Returns:
 Notes:
*****************************************************************************/

void set65XXRcvrParams(P65XX_RCVR_PARAMS *params, int rcvr)
{
    params->decimation_rate = DECIMATION_RATE;
    params->data_mode = DATA_MODE;
    params->data_source = DATA_SOURCE;
    *((unsigned long *)&params->lo) = LO;
    *((unsigned long *)&params->input_clock_rate) = INPUT_CLOCK_RATE;
    params->packing_mode = P65XX_UNPACKED;
    params->data_transfer = P65XX_FIFO;
    params->data_order = P65XX_RECEIVER_ORDER;
    params->rcvr_select = rcvr & 0x1;
    params->tag_enable = P65XX_TAG_OFF;
    params->flip_spectrum = P65XX_FLIP_OFF;
    params->offset_spectrum = P65XX_OFFSET_ON;
#if P6509
    params->filter_select = P6509_80_PERCENT;
    params->rcvr_sync_mode = P6509_GC_SS_OFF | P6509_GC_LD_FREQ;
#endif
#if P6508
    params->wait_ram_full = P65XX_WAIT_RAM_FULL_ON;
#endif
}
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

A.6 main65XX.c - VMEBus only (continued)

A.6.2 set65XXInputParams Function- VMEBus only

```
/*****************************************************************************
 Function: set65XXInputParams
 Description: This routine initializes the Input Channel Parameter Table for
              a specified input channel.
 Inputs:  params - Pointer to a P65XX_INPUT_PARAMS table entry.
          input - Input Channel being setup
 Outputs:
 Returns:
 Notes:
*****************************************************************************/

void set65XXInputParams(P65XX_INPUT_PARAMS *params, int input)
{
  params->threshold_level = P65XX_THRSHLD_DETECT_OFF;
  params->threshold_cross_cnt = P65XX_CROSS_CNT_2;
#if P6508
  params->input_reset_mode = P6508_SW_RESET_IN_SRC | P6508_SW_RESET_OFF |
                             P6508_RESET_OUT_DIS;
#endif
#if P6509
  params->input_sync_mode = P6509_SW_SYNC_SRC | P6509_SW_AS_OFF |
                            P6509_SW_SS_OFF;
#endif
}
```

A.6.3 main65XX Function - VMEBus only

```
/*****************************************************************************
 Function: main65XX

 Description:  This program initializes the specified receivers and
               acquires data.
*****************************************************************************/

main65XX()
{
  register int             i;
  register int             j;
  unsigned long            cmdParams[2];
  unsigned long            status;
  int                      *samples;
  P65XX_REG_ADDR           p65xxReg;
  P65XX_RCVR_PARAMS        p65xxRcvrParams[P65XX_NUM_RCVRS];
  P65XX_INPUT_PARAMS       p65xxInputParams[P65XX_NUM_INPUTS];
  unsigned int             handle[P65XX_NUM_RCVRS] = {0, 0, 0, 0, 0, 0, 0, 0};
  unsigned long            returnData[2];
  unsigned long            numRData;
  unsigned long            a_rcvr;
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

A.6 main65XX.c - VMEBus only (continued)

A.6.3 main65XX Function - VMEBus only (continued)

```
  unsigned long            num_samples;
  char                     filename[20];
  FILE                     *outfile;
  volatile long            *fifo;
  volatile unsigned long   *harris;
  unsigned long vmeBase;   /* VMEbus A24 & A32 memory pointer */
  unsigned long a16Base;   /* VMEbus A16 memory pointer       */

  /* Map Vme Address to Local Address */

  sysBusToLocalAdrs(P65XX_VME_ACCESS_MODE, (char *)P65XX_VME_BASE_ADDR,
                    (char **)&vmeBase);
  printf("VMEBase = %lx\n", vmeBase);

  /* Map A16 Address to Local Address */

  sysBusToLocalAdrs(VME_AM_SUP_SHORT_IO, (char *)P65XX_A16_BASE_ADDR,
                    (char **)&a16Base);
  printf("A16Base = %lx\n", a16Base);

  /* Set 65XX Board's Base Address */

  VxWSet65XXBaseAddress(a16Base, P65XX_VME_BASE_ADDR, P65XX_VME_ACCESS_MODE);
#if INTERRUPT

  /* Set 65XX Board's IRQ Level and Vector */

  VxWSet65XXIRQLevelVector(a16Base, P65XX_INTR_LEVEL, P65XX_INTR_VECTOR);

  /* Set 65XX Board's IRQ Level and Vector */

  if (intConnect(INUM_TO_IVEC(P65XX_INTR_VECTOR), fifoFullIntr) != OK)
  {
     printf("STATUS: INTERRUPT ERROR cant connect int");
     exit(1);
  }

  /* Set Fifo Full Interrupt flag to false */

  FifoFullIntr = 0;

  /* Turn on the Interrupt */

  sysIntEnable(P65XX_INTR_LEVEL);
#endif


/* Initialize pointers to registers */

  InitRegAddr(vmeBase, &p65xxReg);
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

A.6 main65XX.c - VMEBus only (continued)

A.6.3 main65XX Function - VMEBus only (continued)

```
/* Close all prior open connections */

  if ((status = VxWSend65XXCommandWait(a16Base, vmeBase,
                P65XX_BUILD_CMD_WAIT(P65XX_CLOSE_ALL_CONNECTIONS, 0), 0, NULL,
                0, &numRData, returnData)) != P65XX_COMMAND_COMPLETE)
  {
     printf("Error Closing All Connections - Status = %lx\n", status);
     exit(1);
  }

  /* Reset all 65XX Registers */

  if ((status = VxWSend65XXCommandWait(a16Base, vmeBase,
       P65XX_BUILD_CMD_WAIT(P65XX_RESET_REGS, 0), 0, NULL, 0, &numRData,
       returnData)) != P65XX_COMMAND_COMPLETE)
  {
     printf("Error Resetting Registers - Status = %lx\n", status);
     exit(1);
  }

/* Initialize Input Channels */

  for (i = 0; i < P65XX_NUM_INPUTS; ++i)
  {

  /* Initialize Input Channel Parameter Data structure */

     set65XXInputParams(&p65xxInputParams[i], i);

 /* Initialize the Input Channel */

     if ((status = VxWSend65XXCommandWait(a16Base, vmeBase,
                P65XX_BUILD_CMD_WAIT(P65XX_INIT_INPUT_CHANNELS, 0),
                sizeof(P65XX_INPUT_PARAMS), (void *)&p65xxInputParams[i], 1 << i,
                &numRData, returnData)) != P65XX_COMMAND_COMPLETE)
     {
        printf("Error Initializing Input %c - Status = %lx\n", 'A' + i, status);
        exit(1);
     }
  }

  /* Open a connection and initialize each receiver  */

  for (i = 0; i < P65XX_NUM_RCVRS; ++i)
  {
     f (RCVR_MASK & (1 << i))
  {
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

A.6 main65XX.c - VMEBus only (continued)

A.6.3 main65XX Function - VMEBus only (continued)

```
   /* Initialize Parameter Data structure */

 set65XXRcvrParams(&p65xxRcvrParams[i], i + 1);

 /* Establish Connection for this receiver */

   if ((status = VxWSend65XXCommandWait(a16Base, vmeBase,
                 P65XX_BUILD_CMD_WAIT(P65XX_OPEN_CONNECTION, 0), 0, NULL,
                 1 << i, &numRData, returnData)) != P65XX_COMMAND_COMPLETE)
   {
      printf("Error Opening Connection Rcvr %d - Status = %lx\n", i + 1, status);
      exit(1);
   }

   /* Save the handle corresponding to this connection */

   handle[i] = *returnData;

   /* Initialize the Receiver */

   if ((status = VxWSend65XXCommandWait(a16Base, vmeBase,
                 P65XX_BUILD_CMD_INTEGER(P65XX_INIT_RCVRS,\ handle[i],
                 P65XX_WAIT_COMPLETE), sizeof(P65XX_RCVR_PARAMS),
                 (void *)&p65xxRcvrParams[i], 1 << i, &numRData,
                 returnData)) != P65XX_COMMAND_COMPLETE)
   {
      printf("Error Initializing Rcvr %d - Status = %lx\n", i + 1, status);
      exit(1);
   }

   /* Save an index into one of the valid receivers data structures */

   a_rcvr = i;
   }
 }

 /* Reset all the Receiver Fifos */

   if ((status = VxWSend65XXCommandWait(a16Base, vmeBase,
                 P65XX_BUILD_CMD_WAIT(P65XX_RESET_FIFO, handle[a_rcvr]) 0, NULL,
                 RCVR_MASK, &numRData, returnData)) != P65XX_COMMAND_COMPLETE)
 {
   printf("Error Resetting Fifos %d - Status = %lx\n", i, status);
   exit(1);
 }
#if INTERRUPT
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

A.6 main65XX.c - VMEBus only (continued)

A.6.3 main65XX Function - VMEBus only (continued)

```
  /* Enable Fifo Full Interrupt on one of the receivers */

  cmdParams[0] =  P65XX_GEN_VME_INTR | P65XX_ENABLE_INTERRUPT |
                  P65XX_MASK_ON_OCCURRENCE | P65XX_EDGE_TRIG_INTR;
  cmdParams[1] =  P65XX_FIFO_FULL_1 << a_rcvr;
  if ((status = VxWSend65XXCommandWait(a16Base, vmeBase,
                  P65XX_BUILD_CMD_WAIT(P65XX_MODIFY_INTR_MASK, 0), 2, cmdParams,
                  0, &numRData, returnData)) !=  P65XX_COMMAND_COMPLETE)
     {
        printf("Error Enabling Interrupt - Status = %lx\n", status);
        exit(1);
     }
#endif

/* Release the Fifos and begin acquisition */

     VxWSend65XXCommand(a16Base, vmeBase, P65XX_BUILD_CMD(P65XX_RELEASE_FIFO,
                        handle[a_rcvr]), 0, NULL, RCVR_MASK);
     if (p65xxRcvrParams[a_rcvr].data_mode == P65XX_COMPLEX) num_samples =
          NUM_CMPLX_SAMPLES * 2;
     else
        num_samples = NUM_REAL_SAMPLES;
        for (i = 0; i < P65XX_NUM_RCVRS; ++i)
        {
           if (RCVR_MASK & (1 << i))
           {
#if POLLING

        /* Wait for Fifo to Fill Up */

           while (*p65xxReg.Stat_Reg_1 & (0x1 << i));
#else

/* Wait for Fifo to Fill Up */

           while (!FifoFullIntr);
#endif

/* Open an ASCII file for storing data */

           sprintf(filename, "rcvr%d.dat", i + 1);
           outfile = fopen(filename, "w+");
           if (outfile == NULL)
           {
              printf("Error opening file %s\n", filename);
              exit(1);
           }
           printf("Saving Rcvr %d Data to rcvr%d.dat...", i + 1, i + 1);
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

A.6 main65XX.c - VMEBus only (continued)

A.6.3 main65XX Function - VMEBus only (continued)

```
        /* Write Fifo Data to File */

           fifo = p65xxReg.Fifo + 0x2000 * i;
           for (j = 0; j < num_samples; ++j)
                fprintf(outfile, "%d\n", *fifo >> 16);

        /* Close File */

        fclose(outfile);
        printf("Done\n");
      }
  }

/* Close All Open Connections */

  if ((status = VxWSend65XXCommandWait(a16Base, vmeBase,
          P65XX_BUILD_CMD_WAIT(P65XX_CLOSE_ALL_CONNECTIONS, 0), 0, NULL, 0,
          &numRData, returnData)) != P65XX_COMMAND_COMPLETE)
  {
     printf("Error Closing All Connections - Status = %lx\n", status);
     exit(1);
  }
#if INTERRUPT

  /* Turn off the Interrupt */

  sysIntDisable(P65XX_INTR_LEVEL);
#endif
  printf("Program Complete\n");
}
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

A.6 main65XX.c - VMEBus only (continued)

A.6.4 fifoFullIntr Function - VMEBus only

```
/*****************************************************************************
 Function: fifoFullIntr
 Description:  This routine servers as an interrupt handler for Fifo Full
               interrupts generated by the 65XX Board
 Inputs:
 Outputs:
 Returns:
 Notes:
*****************************************************************************/
void fifoFullIntr()
{
/* Set Fifo full Interrupt Flag */
          FifoFullIntr = 1;
}
```

A.7 cmdvxw.h - VMEBus only

```
/*****************************************************************************
*
*   File :  cmdvxw.h
*
*   Description:  This file contains function prototypes for interfacing
*                 a Single Board Computer running vxWorks to a 65XX
*                 Digital Receiver Board.
*
*   $Revision:$   $Date:$
*
*   $History:$
*
*****************************************************************************/

#ifndef CMDVXW_H
#define CMDVXW_H
unsigned long VxWSend65XXCommandWait(unsigned long a16Base, unsigned long
                                     vmeBase, unsigned int command, unsigned int
                                     num_params, void *cmdParams, unsigned int
                                     input_rcvr_mask, unsigned long *numRData,
                                     unsigned long *returnData);
void VxWSend65XXCommand(unsigned long a16Base, unsigned long vmeBase,
                        unsigned int command, unsigned int num_params,
                        void *cmdParams, unsigned int input_rcvr_mask);
void VxWSet65XXBaseAddress(unsigned long a16Base, unsigned long vmeBase,
                           unsigned int accessMode);

void VxWSet65XXIRQLevelVector(unsigned long a16Base, unsigned long irqLevel,
                              unsigned long irqVector);
#endif
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

A.8 cmdvwx.c - VMEBus only

```
/*************************************************************************
*
*   File :  cmdvxw.c
*
*   Description:  This file contains routines for interfacing a Single Board
*                 Computer running vxWorks to a 65XX Digital Receiver Board.
*
*   $Revision:$   $Date:$
*
*   $History:$
*
*************************************************************************/

#include "vxWorks.h"
#include "sysLib.h"
#include "msgQLib.h"
#include "iv.h"
#include "vme.h"
#include "taskLib.h"
#include "usrLib.h"
#include "65xxcmd.h"
#include "cmdvxw.h"
```

A.8.1 VxWSendCommandWait Function - VMEBus only

```
/*************************************************************************
 Function: VxWSendCommandWait

 Description:This routine generates a command to the 65XX Board via the
             VMEBus Interface and waits for a response.
 Inputs:  a16Base - A16 Base Address of 65XX Board
          vmeBase - VME Base Address of 65XX Board
          command - Command being sent
          num_params - The number of parameters which must follow command
          cmdParams - Pointer to the command parameters
          input_rcvr_mask - Input Channel or Receiver mask
          numRdata - Pointer to store the number of returned data
          returnData - Buffer pointer to store the return data
 Outputs: numRdata - Number of returnned data
          returnData - Returned Data
 Returns: Status - Status of the Command
 Notes:
*************************************************************************/
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

A.8 cmdvwx.c - VMEBus only (continued)

A.8.1 VxWSendCommandWait Function - VMEBus only (continued)

```
unsigned long VxWSend65XXCommandWait(unsigned long a16Base, unsigned long vmeBase,
                                     unsigned int command, unsigned int num_params,
                                     void *cmdParams, unsigned int input_rcvr_mask,
                                     unsigned long *numRData, unsigned long *returnData)
{
  unsigned int    i;
  unsigned int    status;
  unsigned long   a16RegBase;
  register volatile unsigned long *vme_command;

/* Initialize pointer to Command location in 65XX DPR */

  vme_command = (unsigned long *)vmeBase + P65XX_VME_COMMAND_ADDR;

/* Write Command */

  *vme_command  = command;

/* Write Input Channel/Receiver Mask */

  *(vme_command + 1) = input_rcvr_mask;

  /* Write the command parameters */

  for (i = 0; i < num_params; ++i)
  *(vme_command + 2 + i) = *((unsigned long *)cmdParams + i);

/* Interrupt the C40 on the 6508 Board */

  *((unsigned long *)a16Base + 2) = 0x1;

/* Wait for Command to Complete */

  while (*vme_command != 0)
  {
  }

/* Read Status and return data */

  status = *(vme_command + 1);
  if ((numRData != NULL) && (returnData != NULL))
  {
     *numRData = status >> 16;
     for (i = 0; i < *numRData; ++i)
     *returnData++ = *(vme_command + 2 + i);
  }
  return(status & 0xffff);
}
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

A.8 cmdvwx.c (continued)

A.8.2 VxWSend65XXCommand Function - VMEBus only

```
/*****************************************************************************
 Function: VxWSend65XXCommand
 Description: This routine generates a command to the 65XX Board via the
              VMEBus Interface.
 Inputs:  a16Base - A16 Base Address of 65XX Board
          vmeBase - VME Base Address of 65XX Board
          command   - Command being sent
          num_params - The number of parameters which must follow command
          cmdParams - Pointer to the command parameters
          input_rcvr_mask - Input Channel or Receiver mask
 Outputs:
 Returns: Command Complete status
 Notes: Returns immediately to caller without waiting command to complete
*****************************************************************************/

void VxWSend65XXCommand(unsigned long a16Base, unsigned long vmeBase,
                        unsigned int command, unsigned int num_params,
                        void *cmdParams, unsigned int input_rcvr_mask)
{
  unsigned int i;
  unsigned int status;
  register volatile unsigned long  *vme_command;

/* Initialize pointer to Command location in 6508 DPR */

  vme_command = (unsigned long *)vmeBase + P65XX_VME_COMMAND_ADDR;

/* Write Command */

  *vme_command  = command;

/* Write Input Channel/Receiver Mask */

  *(vme_command + 1) = input_rcvr_mask;

/* Write the command parameters */

  for (i = 0; i < num_params; ++i)
  *(vme_command + 2 + i) = *((unsigned long *)cmdParams + i);

/* Interrupt the C40 on the 65XX Board */

  *((unsigned long *)a16Base + 2) = 0x1;
}
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

A.8 **cmdvwx.c** (continued)

A.8.3 VxWSet65XXBaseAddress Function - VMEBus only

```
/*****************************************************************************
 Function: VxWSet65XXBaseAddress
 Description:  This routine sets the 65XX's A32/A24 Base Address.
 Inputs:  a16Base - A16 Base Address of 65XX Board
          vmeBase - VME Base Address of 65XX Board
          accessMode - Address Modifier
 Outputs:
 Returns:
 Notes:
*****************************************************************************/

void VxWSet65XXBaseAddress(unsigned long a16Base, unsigned long vmeBase,
                           unsigned int accessMode)
{
  unsigned long *a16 = (unsigned long *)a16Base;
  if ((accessMode & 30) == 0x30                    /* If A24   */
     *(a16 + 1) = ((((vmeBase >> 19) & 0x1e) | 0x20) << 16);
  else                                             /* Else A32 */
     *(a16 + 1) = (((vmeBase >> 16) & 0xffc0) << 16);
}
```

A.8.4 VxWSet65XXIRQLevelVector Function - VMEBus only

```
/*****************************************************************************
 Function: VxWSet65XXIRQLevelVector
 Description: This routine sets the 65XX's Interrupt Level and Vector
 Inputs:  a16Base - A16 Base Address of 65XX Board
          irqLevel  - Interrupt Level
          irqVector  - Interrupt Vector
 Outputs:
 Returns:
 Notes:
*****************************************************************************/

void VxWSet65XXIRQLevelVector(unsigned long a16Base, unsigned long irqLevel,
                              unsigned long irqVector)
{
  unsigned long *a16 = (unsigned long *)a16Base;
  *a16 = (*a16 & ~0x03ff0000) | (irqLevel << 24) | (irqVector << 16);
}
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

This page is intentionally blank

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Appenedix B: Graychip GC1011A Data Sheet

B.1 Introduction

Included in its entirety, beginning on the page after next, is the manual for the Graychip GC1011A digital Down Converter. Eight of these devices are used on the Model 6509.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

This page is intentionally blank

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

GRAYCHIP
DSP CHIPS AND SYSTEMS
(415) 323-2955
FAX (415) 323-0206
2314 Ramona St. Palo Alto, CA. 94301

GC1011A

DIGITAL RECEIVER CHIP

DATASHEET

Revised

November 24, 1992

This datasheet contains information which may be changed at any time without notice.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

This page is intentionally blank

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

GC1011A DATASHEET

1.0 FUNCTIONAL DESCRIPTION

Fabricated in 1 micron CMOS technology, the GC1011A chip is an all digital receiver which can extract narrow band signals from wide band digitized sources. At full rate operation (70 MHz input rate), the input bandwidth can be up to 35 MHz wide. Any signal within the input bandwidth can be down-converted to zero frequency, low pass filtered, and output at a reduced sample rate. The narrow band output can be formatted as either a complex data stream, or a real data stream. In the complex output mode the output sample rate ranges from $(1/64)^{th}$ to $(1/65,536)^{th}$ of the input rate in steps of (1/4N) where N is an integer ranging from 16 to 16,384. The output sample rate is doubled in the real output mode.

The user may select a standard output filter which has a passband covering 80% of the output bandwidth and has more than 70 dB of out-of-band rejection, or a wider, 90% passband filter which has 50 dB of out-of-band rejection.

The 28 bit accumulator in the chip's digital oscillator circuit provides a tuning accuracy equal to the input clock rate divided by 2^{28}. With a 60 MHz clock rate the tuning resolution is 0.2 Hz giving a tuning accuracy of +/- 0.1 Hz. The phase noise in the oscillator is low enough to provide a spur free dynamic range of over 75 dB. All arithmetic following the oscillator is performed to sufficient accuracy to provide over 90 dB of noise free dynamic range. This allows the chip to generate and preserve up to 20 dB of narrow band processing gain. The gain of the output signal can be adjusted in 6 dB steps.

The chip's output circuit allows the user to select a real or complex data output format, to select spectral inversion, or to offset the output spectrum by half of the output sample rate. The filtered signal is output in a bit-serial format. The user has control over the bit-serial format including its rate, its bit-strobe format and its frame-strobe format.

On chip diagnostics are provided to simplify system debug and maintenance.

The chip receives configuration and control information over a microprocessor compatible bus consisting of an 8 bit data I/O port, a 4 bit address port, a read/$\overline{\text{write}}$ bit, and a control select strobe.

The GC1011A chip replaces the GC1011 chip. The new features include improved real output formatting and a symmetric rounding algorithm which removes unwanted DC rounding bias from the output.

1.1 KEY FEATURES

- 70 million samples per second input rate
- 0.1 Hz tuning resolution
- >70 dB dynamic range
- Programmable output bandwidth
- 12 bit inputs, 16 bit outputs
- Real or complex output formats
- Built in strobe/sync generator
- Microprocessor interface for control, output, and diagnostics
- Power down mode
- Built in diagnostics
- 750 mW power at 50 MHz, 5 volts
- 200 mW at 30 MHz, 3.3 volts
- Small, 44 pin Leaded chip carrier package

This document contains information which may be changed at any time without notice

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

1.2 BLOCK DIAGRAM

A block diagram illustrating the major functions of the chip is shown in Figure 1

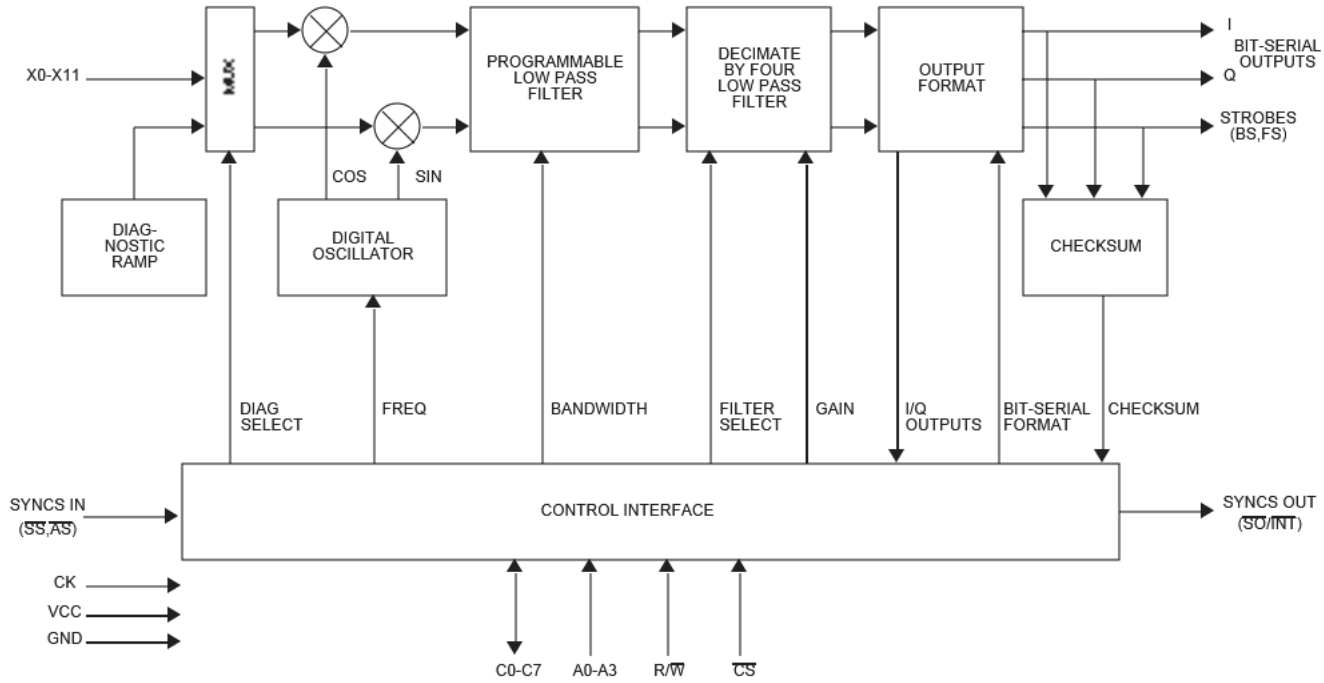


Figure 1. GC1011A BLOCK DIAGRAM

Each of these functions are described below.

1.3 CONTROL INTERFACE

The control interface performs four major functions: It allows an external processor to configure the chip, it allows an external processor to capture and read output samples from the chip, it allows an external processor to perform diagnostics, and it generates internal synchronization strobes.

The chip is configured by writing control information into 8 bit control registers within the chip. The contents of these control registers and how to use them are described in Section 3. The registers are written to or read from using the **C[0:7]**, **A[0:3]**, **R/$\overline{W}$**, and **$\overline{CS}$** pins. Each control register has been assigned a unique address within the chip. An external processor (a microprocessor, computer, or DSP chip) can write into a register by setting **A[0:3]** to the desired register address, setting the **R/$\overline{W}$** pin low, setting **D[0:7]** to the desired value and then pulsing **$\overline{CS}$** low.

To read from a control register the processor must set **A[0:3]** to the desired address, set **R/$\overline{W}$** high, and then set **$\overline{CS}$** low. The chip will then drive **C[0:7]** with the contents of the selected register. After the processor has read the value from **C[0:7]** it should set **$\overline{CS}$** high. The **C[0:7]** pins are turned off (high impedance) whenever **$\overline{CS}$** is high or **R/$\overline{W}$** is low. The chip will only drive these pins when **$\overline{CS}$** is low and **R/$\overline{W}$** is high.

This document contains information which may be changed at any time without notice

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Control register addresses 12, 13, 14, and 15 are reserved to allow an external processor to read output samples from the chip. Addresses 12 and 13 are the I-registers which store the16 bit real part of the output sample. Addresses 14 and 15 are the Q-registers which store the imaginary part. In the real mode the I registers store the even-time output samples and the Q registers store the odd-time output samples. Output ready and missed flags are provided in control register 10 in order to synchronize the storing and reading of the output samples. An interrupt output pin is also provided on the chip which can be used to interrupt the external processor when a new sample is ready. See the description of control register 10 in Section 3.7 for more details. The setup, hold and pulse width requirements for control read or write operations are given in Section 4.4.

Checksums are read from the chip during diagnostics using address 11. More details on the diagnostic modes is given in Section 1.9.

The control interface also generates the chip's internal sync strobes. The user may select to synchronize the chip using an external sync strobe ($\overline{\mathbf{SS}}$), or use the chip's internal sync counter. The internal sync counter can be synchronized to $\overline{\mathbf{SS}}$, or left to free run (See **SS_OFF** in Section 3.2). The period of the internal sync counter can be either 256 clocks or 2^{20} clocks. The 256 clock period is intended to be used for chip test purposes only. The internal sync counter is used during diagnostics to clear the data paths and strobe the checksum generator. The internal sync counter can also be used to periodically re-synchronize all of the counters in the chip during normal operating modes if the chip's decimation ratio is set to a power of 2. Except for diagnostics the sync counter cannot be used for non-power of 2 decimation modes.

1.4 DIGITAL OSCILLATOR

The digital oscillator generates sine and cosine sequences which are used to mix the desired signal down to zero frequency. The digital oscillator contains a 28 bit frequency register, a 28 bit frequency accumulator, and a sine-cosine generator. The tuning frequency of the oscillator is set by loading a 28 bit frequency word from the control registers into the frequency register. If the frequency register is set to the word **FREQ**, then the tuning frequency will be: $\text{Frequency} = \frac{\text{Sample Rate}}{2^{28}}\mathbf{FREQ}$.

The frequency word **FREQ** is stored into the control registers at control addresses 0,1,2 and 3. The 28 bit word is then transferred into the frequency register using one of the following methods:

(1) The frequency register is always loading (the frequency changes immediately as the frequency word is loaded into the control registers).

(2) The frequency register is loaded when the user sets a control register bit.

(3) The frequency register is synchronously loaded when the accumulator sync strobe ($\overline{\mathbf{AS}}$) goes low.

(4) The frequency register is synchronously loaded when the system sync strobe ($\overline{\mathbf{SS}}$) goes low.

See Section 3.2 for more details on the frequency load modes.

The 28 bit frequency word is accumulated in the 28 bit frequency accumulator. The frequency accumulator will normally free run, but can be synchronously cleared by either the system sync ($\overline{\mathbf{SS}}$) or the

This document contains information which may be changed at any time without notice

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

accumulator sync (**$\overline{\text{AS}}$**). The accumulator clear modes are controlled by bits in control register 4. See Section 3.2 for details.

The upper 13 frequency accumulator bits are used to generate the oscillator's sine and cosine outputs. These sines and cosines are generated to 12 bit accuracy. The spectral purity of the oscillator is illustrated in the following figures.

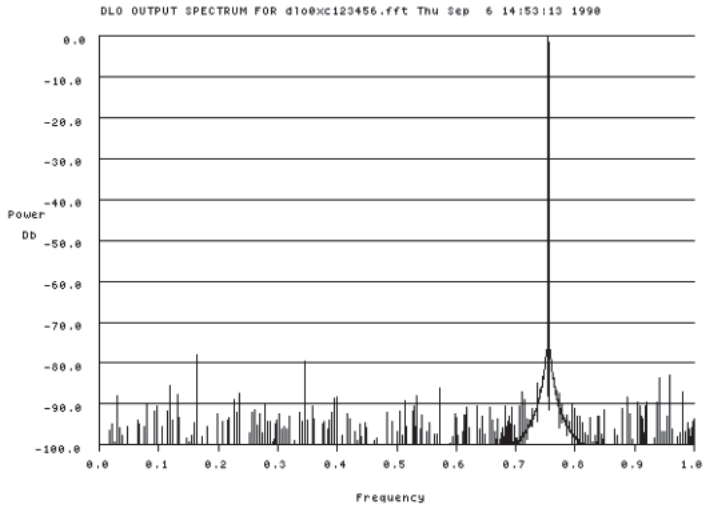


Figure 2. DLO SPECTRUM: FREQ = (HEX) 1234567

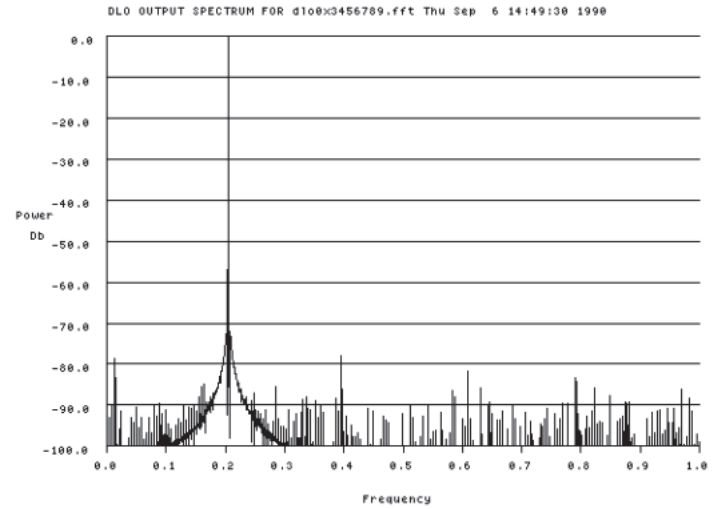


Figure 3. DLO SPECTRUM: FREQ = (HEX) 3456789

This document contains information which may be changed at any time without notice

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

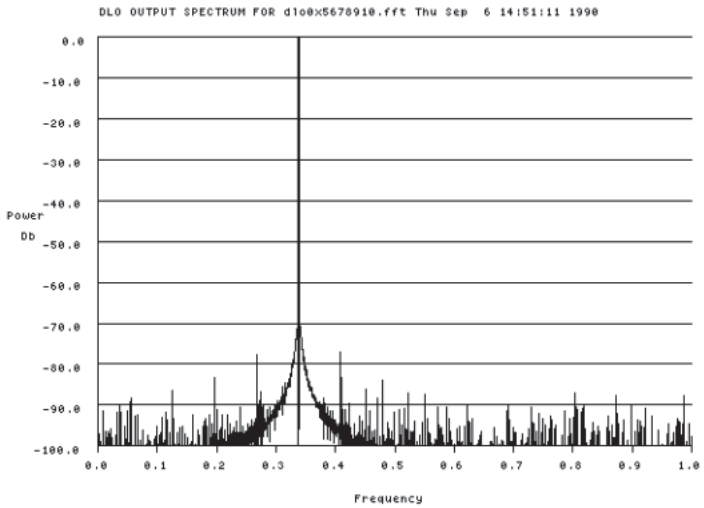


Figure 4. DLO SPECTRUM: FREQ = (HEX) 5678910

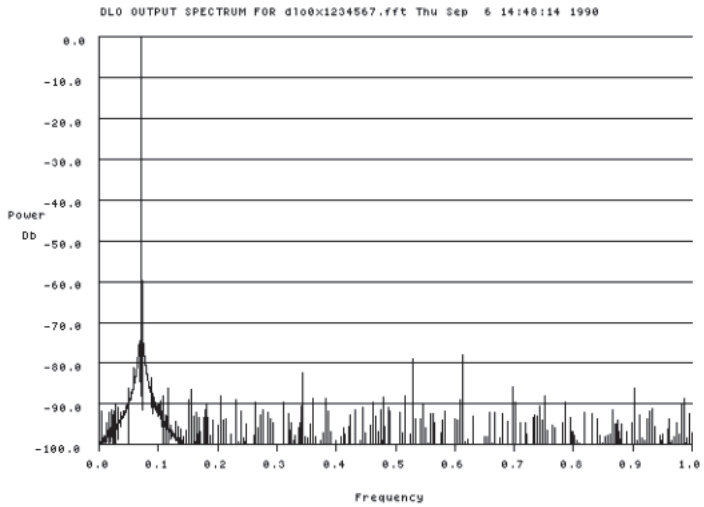


Figure 5. DLO SPECTRUM: FREQ = (HEX) c1234567

The oscillator's peak spur levels, as can be seen from these plots, are well below -75 dB.

1.5 MIXER

The mixer multiplies the 12 bit input samples by the 12 bit sine and cosine values coming from the digital oscillator. An input signal at the oscillator's tuning frequency will be centered at zero frequency after passing through the mixer.

This document contains information which may be changed at any time without notice

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

1.6 PROGRAMMABLE LOW PASS FILTER

The mixer's output is filtered to isolate the desired signal using a programmable bandwidth low pass filter. The filter reduces the sample rate by a factor of **D**, where **D** is a user specified integer ranging from 16 to 16,384. The value of **D** is stored in control registers 8 and 9. The out-of-band rejection of the filter is over 72 dB.

1.7 DECIMATE BY FOUR FILTER

The programmable bandwidth filter is followed by a 64 tap filter which, in the complex mode, reduces the sample rate by an additional factor of 4, giving an overall bandwidth (and sample rate) reduction of 4**D**. In the real mode the final filter only reduces the sample rate by a factor of two. The user may select a choice of two spectral responses for this filter. The standard response has a 3 dB passband which covers 80% of the output Nyquist rate and has an out-of-band rejection of over 70 dB. The optional response has a 3 dB passband covering 90% of the Nyquist rate, but only 50 dB of out-of-band rejection. The spectral response of these filters is shown in the following figures.

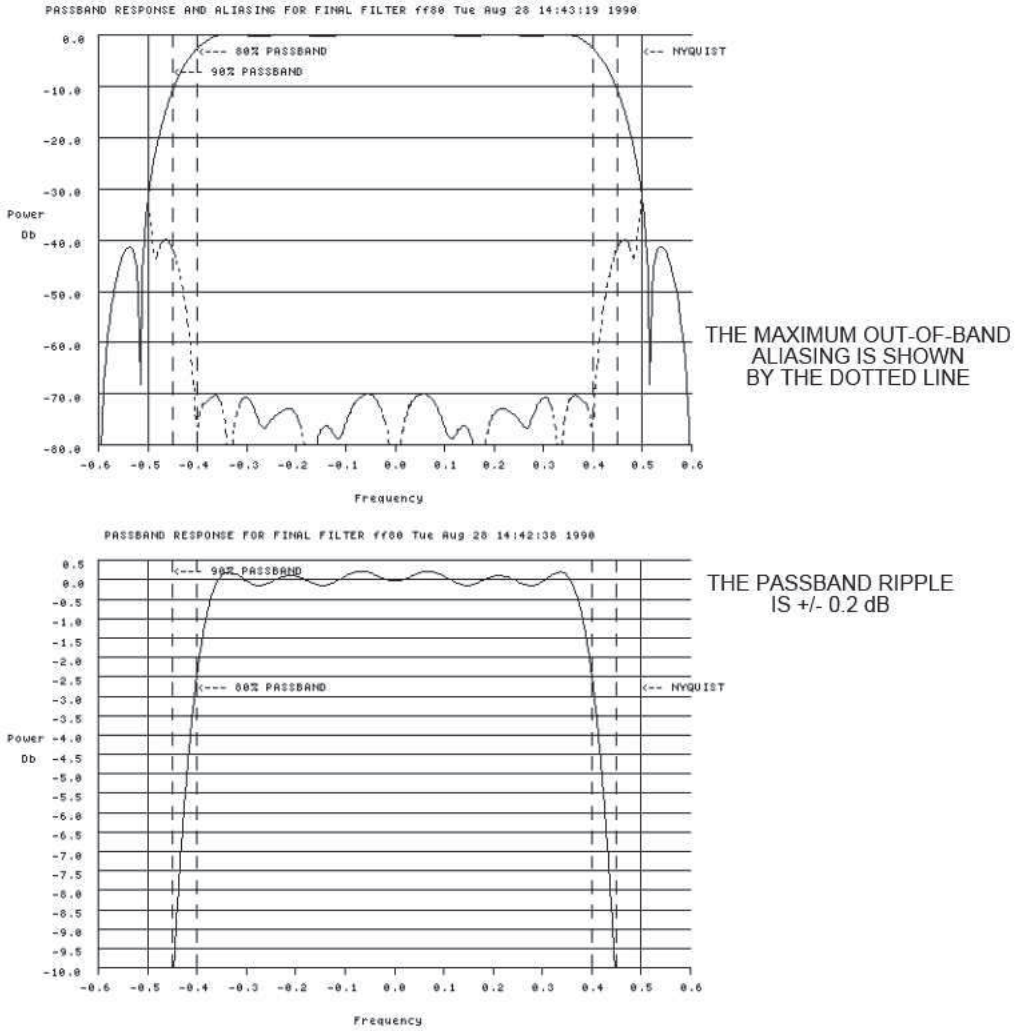
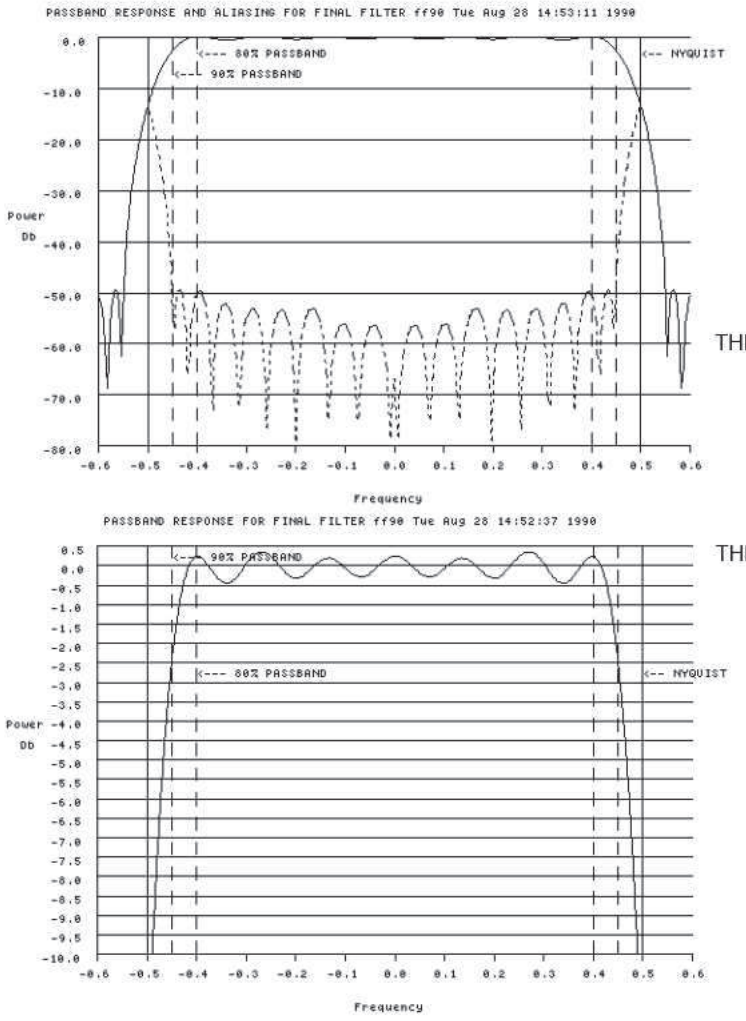


Figure 6. FILTER RESPONSE: 80% FILTER

This document contains information which may be changed at any time without notice

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com



THE MAXIMUM OUT-OF-BAND ALIASING IS SHOWN BY THE DOTTED LINE

THE PASSBAND RIPPLE IS +/- 0.4 dB

Figure 7. FILTER RESPONSE: 90% FILTER

This document contains information which may be changed at any time without notice

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

1.8 OUTPUT FORMATTING

The output format circuit adjusts the gain of the filter output, optionally flips the output spectrum, optionally offsets the spectrum by one-fourth the Nyquist rate, optionally converts the complex output stream to a real one at twice the rate, and rounds the samples to 16 bits. The rounded outputs are passed to the control interface where they can be read by an external processor. They are also passed through a parallel to serial conversion circuit and output as bit-serial streams. Separate bit-serial outputs are provided for the **I** and **Q** halves of the complex output.

The bit-serial output has several programmable features. The user can select:

(1) The rate of the bit-serial stream. The bit rate is programmable to be a power of two division of the clock.

(2) MSB first or LSB first orientation. The 16 bit data is normally output LSB first, but can optionally be output MSB first.

(3) Multiplexed **I**/**Q** format. The user can have the **I** and **Q** outputs multiplexed together so that the bit streams contain the 16 bit **I** sample followed by the 16 bit **Q** sample. The **I** and **Q** outputs are identical in this mode.

(4) The format of the bit strobe. A bit strobe is provided as a clock for the bit-serial stream. The bit strobe can be selected to have a 50% duty cycle, or to have a pulse width equal to a system clock period. The user can also select the active high or active low polarity of the bit strobe. The clock pulse mode is suitable to be used as a clock enable signal. The bit strobe is a constant high or low level (depending upon the mode and polarity selected) when the bit rate is set to be equal to the clock rate.

(5) The format of the frame strobe. A frame strobe is provided at the start of each output frame. The strobe is either one bit-period wide synchronous with the first bit of the frame, or it is active during the clock cycle before the start of each frame. The active high or low polarity of the strobe is programmable. The frame rate is equal to the complex output rate.

The programmable bit rate allows the user to select the number of times that each 16 bit output sample repeats during a frame. If the frame period is 64 clocks (i.e., the chip is decimating by a factor of 64), then the same 16 bit output sample will be repeated 4 times when the bit rate is equal to the system clock rate. If the bit rate is set to one-half, then the sample will only repeat twice. If the bit rate is set to one-fourth, then there will be exactly one output sample per frame. The higher bit rates allow one to multiplex the outputs from multiple GC1011A chips into a single time division multiplexed (TDM) stream. If the bit rate is set so that each output sample is repeated four times in a frame, then the outputs from four chips can be multiplexed into a TDM stream by selecting the first chip's output for the first 16 bit-strobes of the frame, selecting the second chip's output for the next 16 strobes, selecting the third chip's output for the next 16 strobes, and then selecting the last chip's output for the final 16 strobes of the frame.

In the real output mode the frame rate is double the complex mode rate and the output signal is shifted in frequency so that it is centered from 0 to $F_O/2$, where F_O is the output sample rate. The I and Q output streams are identical in the real mode. For compatibility with system designs using the older GC1011 chip see the REAL_FS bit described in Section 3.7.

The output format circuitry is synchronized by the $\overline{\mathbf{SS}}$ input sync. This allows one to synchronize the output frame timing of multiple GC1011A chips.

This document contains information which may be changed at any time without notice

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

1.9 DIAGNOSTICS

An input ramp generator, a sync period generator, and a checksum generator are provided on the chip in order to run diagnostic tests. Diagnostics are performed by turning on the ramp generator, enabling the diagnostic syncs, letting the chip operate for at least 4 sync periods, reading the checksum and comparing it to its predicted value. A new checksum is generated every sync period. The input ramp sequence is the same for every sync period and the chip is re-initialized at the beginning of each sync period so that each checksum should be the same once the chip's data path has been flushed. The chip requires at least 3 sync periods to flush, so the fourth and following checksums should be valid. The test is then repeated for several different tuning frequencies, decimation settings, and output modes.

The sync period is 2^{20} clocks, or approximately 1 million clock cycles, so four sync periods will be about 4 million clocks. This represents a delay of less than 15 milliseconds for a clock rate of 60 MHz.

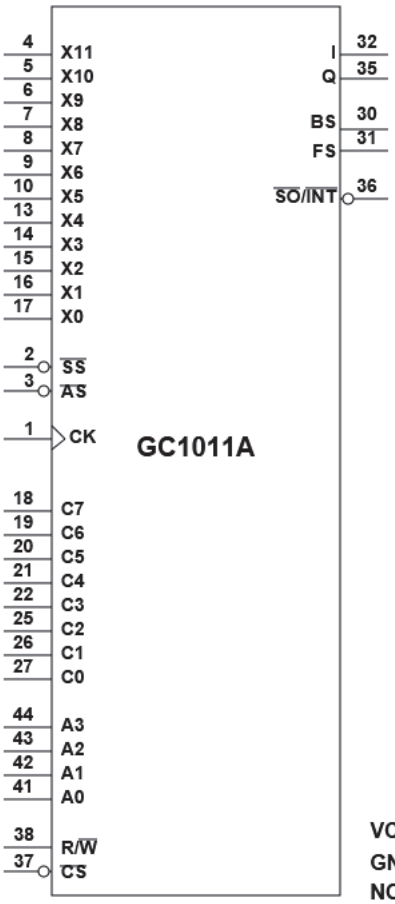
The following table lists the expected checksums for four test configurations. All values are in HEX.

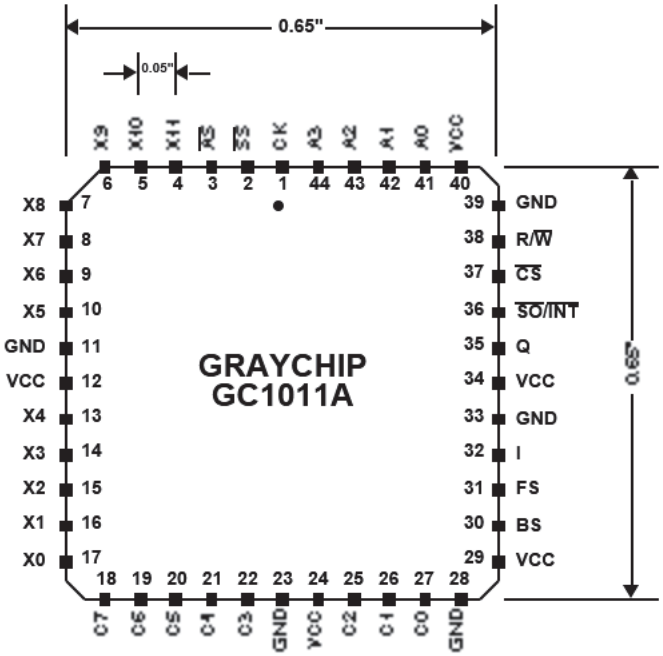
CONTROL REGISTER	TEST 1	TEST 2	TEST 3	TEST 4
FREQ (REG 0,1,2,3)	0000101	0F0F0F0	55AA55A	AA55AA5
SYNC MODE (REG 4)	A9	A9	A9	A9
OUTPUT MODE (REG 5)	01	51	F2	0C
FILTER MODE (REG 6)	50	66	7D	4B
GAIN (REG 7)	0	1	2	3
DECIMATION (REG 8,9)	000F	0020	0F30	30C1
EXPECTED CHECKSUMS (GC1011A) (REG 11)	32	9E	22	52
EXPECTED CHECKSUMS (GC1011) (REG 11)	96	EB	9A	A6

This document contains information which may be changed at any time without notice

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

2.0 PIN DESCRIPTIONS





44 PIN J-LEAD CHIP CARRIER

(CALL GRAYCHIP FOR ALTERNATE PACKAGE AVAILABILITY)

VCC PINS: 12, 24, 29, 34, 40

GND PINS: 11, 23, 28, 33, 39

NOTE: 0.01 to 0.1 µf DECOUPLING CAPACITORS SHOULD BE PLACED AS CLOSE AS POSSIBLE TO PINS 11,12 AND PINS 33,34

SIGNAL	DESCRIPTION
X[0:11]	**INPUT DATA.** *Active high* The 12 bit two's complement input samples. New samples are clocked into the chip on the rising edge of the clock. The input data rate is assumed to be equal to the clock rate.
CK	**CLOCK INPUT.** *Active high* The clock input to the chip. The **X**, **SS**, and **AS** signals are clocked into the chip on the rising edge of this clock. The **I**, **Q**, **BS**, **FS**, and **SO** signals are clocked out on the rising edge of this clock.
SS	**SYSTEM SYNC**. *Active low* The sync input to the chip. All timers, accumulators, and control counters are, or can be, synchronized to **SS**. Bits in control register 4 (see Section 3.2) determine the operation of **SS**. This sync is clocked into the chip on the rising edge of the clock.

This document contains information which may be changed at any time without notice

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

$\overline{AS}$ **ACCUMULATOR SYNC**. *Active low*
The accumulator sync is provided to synchronously change tuning frequencies. This sync can be used to load a new tuning frequency into the frequency register and/or to clear the frequency accumulator. This signal is clocked into the chip on the rising edge of the clock.

I **IN-PHASE OUTPUT STREAM**. *Active high*
The I part of each complex output sample is output as a bit serial stream on this pin. The bits are clocked out on the rising edge of the clock.

Q **QUADRATURE OUTPUT STREAM**. *Active high*
The Q part of each complex output sample is output as a bit serial stream on this pin. The bits are clocked out on the rising edge of the clock.

BS **BIT STROBE**. *Programmable active high or low level*
This strobe is output synchronous with the **I** and **Q** bit streams. The strobe occurs once per bit and is either one clock wide or has a 50% duty cycle. The high/low polarity of the strobe is programmable. See Section 3.3 for details.

FS **FRAME STROBE**. *Programmable active high or low level*
This strobe marks the beginning of an output frame. The strobe is either one clock cycle wide, or is the width of a data bit at the selected bit rate. The high/low polarity of this signal is programmable. See Section 3.3 for details.

$\overline{SO}/\overline{INT}$ **SYNC OUT OR INTERRUPT OUT**. *Active low*
This signal is either a sync output or is an interrupt output depending upon a bit in control register 10. As a sync output, the signal is a delayed version of the input system sync **$\overline{SS}$**, or is the internally generated sync which has a period of 2^{20} clocks, or is a one-shot sync strobe. As an interrupt output, the signal is the READY flag from control register 10. This flag goes active when a new output sample is ready in control registers 12, 13, 14, and 15.

C[0:7] **CONTROL DATA I/O BUS**. *Active high*
This is the 8 bit control data I/O bus. Control register data is loaded into the chip or read from the chip through these pins. The chip will only drive these pins when **$\overline{CS}$** is low and **R/$\overline{W}$** is high.

A[0:3] **CONTROL ADDRESS BUS**. *Active high*
These pins are used to address the 16 control registers within the chip. Each of the 16 control registers within the chip are assigned a unique address. A control register can be written to or read from by setting **A[0:3]** to the register's address.

R/$\overline{W}$ **READ/$\overline{WRITE}$ CONTROL**. *High for read, low for write*
This pin determines if the control bus cycle is a read or write operation. The pin is high for a read and is low for a write.

$\overline{CS}$ **CONTROL STROBE**. *Active low*
This control strobe enables the read or write operation. The contents of the register selected by **A[0:3]** will be output on **C[0:7]** when **R/$\overline{W}$** is high and **$\overline{CS}$** is low. If **R/$\overline{W}$** is low when **$\overline{CS}$** goes low, then the selected register will be loaded with the contents of **C[0:7].**

This document contains information which may be changed at any time without notice

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

3.0 CONTROL REGISTERS

The chip is configured and controlled through the use of 16 eight bit control registers. These registers are accessed for reading or writing using the control bus pins ($\overline{\mathbf{CS}}$, **R/$\overline{\mathbf{W}}$**, **A[0:3]**, and **C[0:7]**) described in the previous section. The register names and their addresses are:

ADDRESS	NAME	ADDRESS	NAME
0	FREQ byte 0	8	Decimation byte 0
1	FREQ byte 1	9	Decimation byte 1
2	FREQ byte 2	10	Output Status
3	FREQ byte 3	11	Checksum
4	Sync mode	12	I-output byte 0
5	Output mode	13	I-output byte 1
6	Filter mode	14	Q-output byte 0
7	Gain	15	Q-output byte 1

The following sections describe each of these registers. The type of each register bit is either R or R/W indicating whether the bit is read only or read/write. All bits are active high.

This document contains information which may be changed at any time without notice

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

3.1 FREQUENCY WORD REGISTERS

Registers 0, 1, 2, and 3 contain the 28 bit frequency tuning word. Bit 0 is the LSB, bit 27 is the MSB.

ADDRESS 0: FREQUENCY BYTE 0

BIT	TYPE	NAME	DESCRIPTION
0-7	R/W	**FREQ**[0:7]	Byte 0 (least significant) of frequency word

ADDRESS 1: FREQUENCY BYTE 1

BIT	TYPE	NAME	DESCRIPTION
0-7	R/W	**FREQ**[8:15]	Byte 1 of frequency word

ADDRESS 2: FREQUENCY BYTE 2

BIT	TYPE	NAME	DESCRIPTION
0-7	R/W	**FREQ**[16:23]	Byte 2 of frequency word

ADDRESS 3: FREQUENCY BYTE 3

BIT	TYPE	NAME	DESCRIPTION
0-3	R/W	**FREQ**[24:27]	4 most significant bits of the frequency word
4-7	R/W	-	unused

If the desired tuning frequency is **F**, then the frequency word should be set to:

$$\mathbf{FREQ} = 2^{28}\mathbf{F}/(\text{clock rate})$$

This document contains information which may be changed at any time without notice

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

3.2 SYNC MODE REGISTER

The sync mode register controls the action of the **$\overline{SS}$** and **$\overline{AS}$** sync strobes and how they affect the chip's internal timers, counters, and accumulators.

ADDRESS 4: Sync Mode Register

BIT	TYPE	NAME	DESCRIPTION
0 (LSB)	R/W	SS_OFF	This bit disables the **$\overline{SS}$** input.
1	R/W	AS_ON	Enables the accumulator sync **$\overline{AS}$**. Normally the frequency accumulator will free run. This bit causes the frequency accumulator to be initialized to the contents of the frequency register when **$\overline{AS}$** goes low. **$\overline{SS}$**, instead of **$\overline{AS}$**, will reset the accumulator if AS_MUX is set and **$\overline{SS}$** is not disabled by SS_OFF.
2	R/W	AS_MUX	Use **$\overline{SS}$** for the accumulator sync. The **$\overline{AS}$** input is ignored and the **$\overline{SS}$** strobe is used in its place when this bit is set and **$\overline{SS}$** is not disabled by SS_OFF. (See AS_ON and AS_FREQ).
3	R/W	LD_FREQ	Load the frequency register in the digital oscillator with the contents of the frequency word registers. If left on, this bit will cause the frequency register to load whenever a frequency word register is changed.
4	R/W	AS_FREQ	Enables the synchronous frequency load mode. When this bit is set and **$\overline{AS}$** goes low, the frequency register will be synchronously loaded with the contents of the frequency control registers. **$\overline{SS}$**, instead of **$\overline{AS}$**, will load the frequency register if AS_MUX is set and **$\overline{SS}$** is not disabled by SS_OFF.
5	R/W	SS_DIAG	Enables diagnostic syncs. This bit routes the internal sync to the checksum generator and to all accumulators and control counters within the chip. This forces the chip to re-initialize at the start of every sync period. The internal sync period will be 2^{20} clocks if SS_MUX is set, otherwise it will be determined by the period of an externally provided **$\overline{SS}$** strobe.
6	R/W	TEST	Shortens the internal sync counter period from 2^{20} clocks to 2^{8} clocks. This mode is used to test chips at the factory.
7 (MSB)	R/W	SS_MUX	Use the sync counter's terminal count strobe for the internal sync instead of the sync input **$\overline{SS}$**. The internal sync is output on the **$\overline{SO}$** pin.

The operation of these control bits are illustrated in Figure 8 on the next page.

This document contains information which may be changed at any time without notice

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

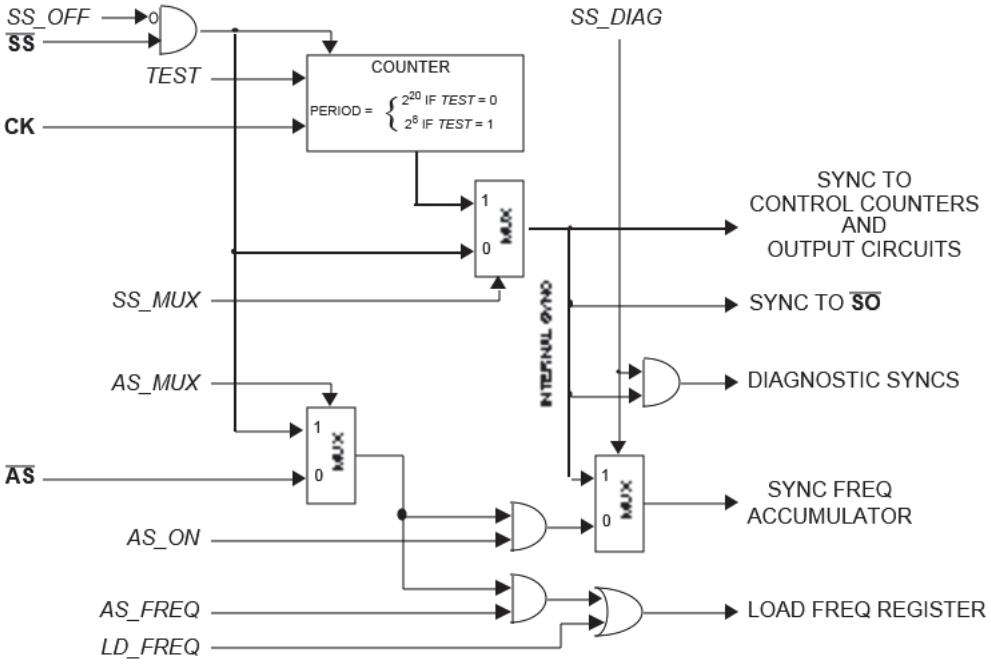


Figure 8. SYNC CONTROLS

This document contains information which may be changed at any time without notice

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

3.3 OUTPUT MODE REGISTER

The output mode register contains the controls for the bit-serial outputs.

ADDRESS 5: Output mode register

BIT	TYPE	NAME	DESCRIPTION
0-3	R/W	**RATE**[0:3]	The bit rate select. The bit serial rate is: Bit rate = $2^{-\mathbf{RATE}}$ (clock rate)
4	R/W	BS_MODE	Changes the mode of **BS**. Normally **BS** pulses high during the clock cycle before a bit transition. This bit changes **BS** so that it is a 50% duty cycle clock with its rising edge in the middle of each bit period
5	R/W	BS_POL	Changes the polarity of **BS**.
6	R/W	FS_MODE	Changes the mode of **FS**. Normally **FS** pulses high during the clock cycle before the start of a new frame. This bit changes **FS** so that it will be high for the first bit period of each new frame.
7	R/W	FS_POL	Changes the polarity of **FS**.

These controls are illustrated in the timing diagrams shown in Figure 9 on the next page. Note that the polarity shown for **BS** and **FS** can be changed using *BS_POL* and *FS_POL*.

This document contains information which may be changed at any time without notice

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

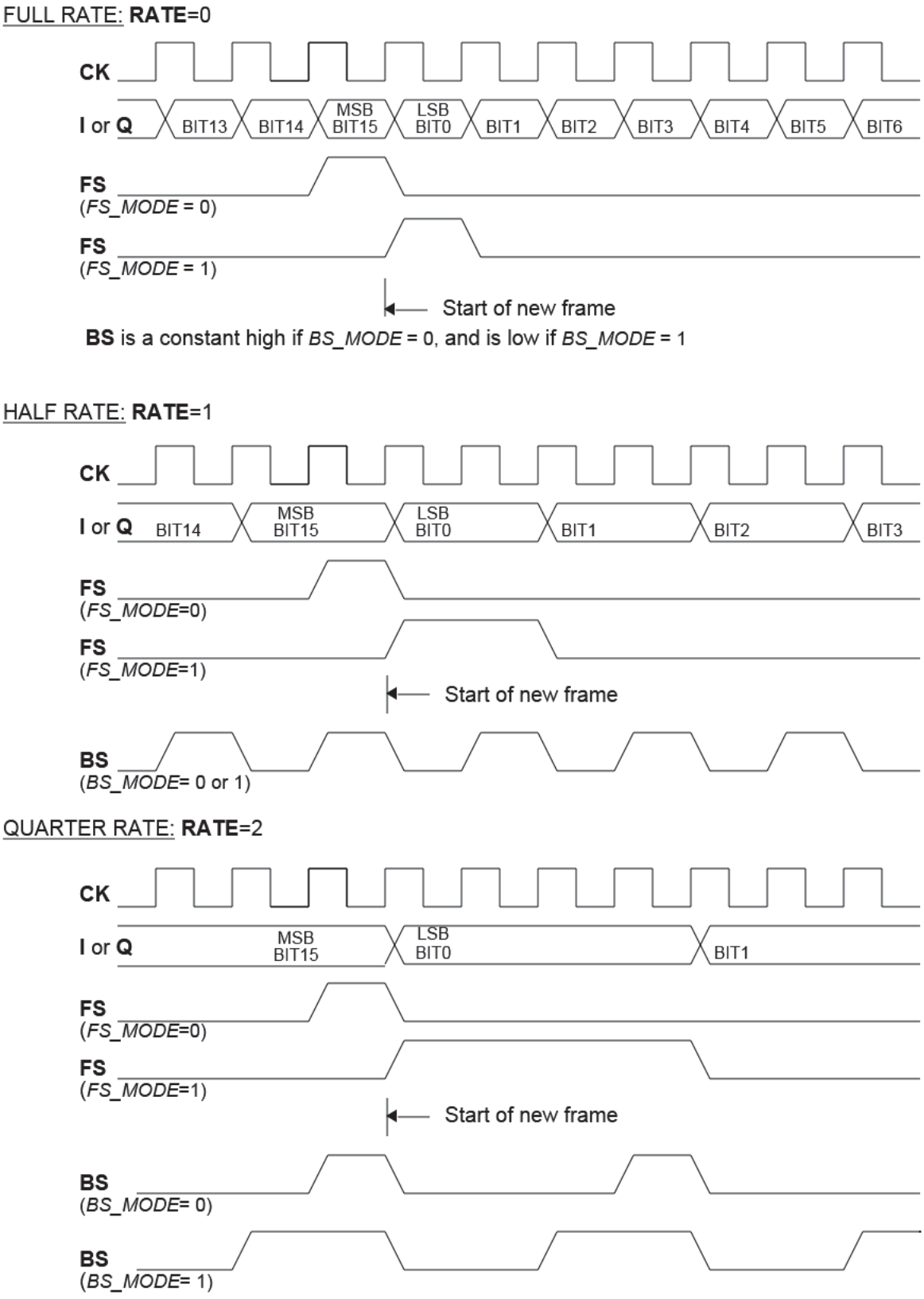


Figure 9. TIMING FOR BIT-SERIAL MODES

This document contains information which may be changed at any time without notice

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

3.4 FILTER MODE REGISTER

This register controls filtering, output formatting and the diagnostic input mode.

ADDRESS 6: Filter Mode Register

BIT	TYPE	NAME	DESCRIPTION
0	R/W	FSEL	Use the 90% filter instead of the 80% filter.
1	R/W	REAL	Use real output format instead of complex. The OFFSET and REAL_FS (see Section 3.7) control bits must also be high for the real mode to work. The output data rate is double the complex data rate in this mode.
2	R/W	OFFSET	Offset the output spectrum. In the complex output mode (REAL=0), this bit forces the output spectrum to be centered at $F_{CK}/2$, where F_{CK} is the output sample rate. This mode is useful for single-sideband AM signals because it moves the lower band edge up to zero frequency where it belongs. The upper half of the spectrum, in the complex mode, will appear as negative frequencies. This mode is mandatory in the real output mode.
3	R/W	FLIP	Flip the output spectrum. This bit inverts the output spectrum. In the complex mode the spectrum is flipped about zero. In the real mode the spectrum is flipped about $F_{CK}/4$, where F_{CK} is the real mode's output sample rate.
4	R/W	IQ_MUX	Multiplex the I and Q outputs together. In this mode the bit -serial output alternates between the 16 bit I sample and the 16 bit Q sample. The I sample is output first in each frame followed by the Q sample. The I and Q samples alternate, 16 bits each, until the end of the frame. The **I** and the **Q** pins output identical bit-streams in this mode.
5	R/W	MSB_MODE	Output MSB first instead of LSB first. Normally the 16 bit samples are output LSB first. The MSB is output first when this bit is set.
6	R/W	DIAG	Use the diagnostic ramp for the input to the chip instead of the **X** input. The ramp counts from -2048 to +2047 and then starts over again.
7	R/W	POWER_DOWN	Allows the $\overline{\textbf{AS}}$ input to gate the chip's internal clock. The clock is always enabled when this bit is low.

This document contains information which may be changed at any time without notice

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

The effect on the output spectrum of the REAL, OFFSET and FLIP bits is illustrated in the following diagram. (F_O is the output sample rate))

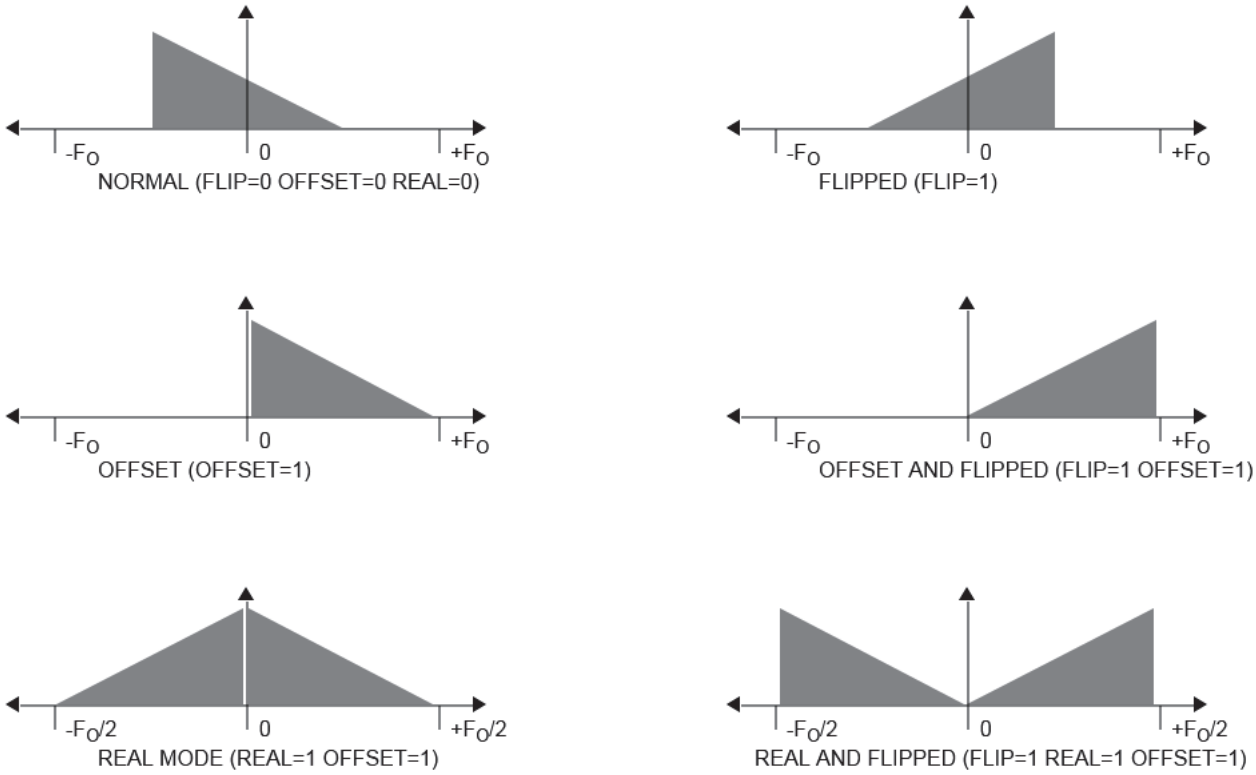


Figure 10. OUTPUT SPECTRAL FORMATS

The POWER_DOWN control bit is used in conjunction with the **$\overline{\text{AS}}$** input pin to decrease the power consumed by the GC1011A chip in a stand-by mode. The power can be reduced by a factor of N (to a minimum of about 1mW) by setting the POWER_DOWN control bit and then pulsing **$\overline{\text{AS}}$** low for one clock cycle out of N. All control registers retain their state during power down, but the chip's **I**, **Q**, **BS**, **FS** and **SO** output signals will not be valid. Note that an internal clock rate (F_{CK}/N) less than 10 Hz may damage the chip.

This document contains information which may be changed at any time without notice

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

3.5 GAIN CONTROL REGISTER

This register sets the output gain.

ADDRESS 7: Gain Control Register

BIT	TYPE	NAME	DESCRIPTION
0-3	R/W	**GAIN[0:3]**	The 4 bit gain word. The gain is equal to $2^{\mathbf{GAIN}}$, allowing gain adjustments in 6 dB increments.
4-7	R/W	-	Unused

The chip has unity gain when the decimation in the programmable low-pass filter has been set to a power of two. If the decimation is not a power of two, then the gain is given by $G=(\mathbf{D}/\mathbf{P})^4$, where **D** is the decimation in the programmable low-pass filter (See Section 3.6) and **P** is the next power of two greater than **D**. For example, if **D** is 30 the gain would be $(30/32)^4=0.772$, or -2.2 dB. If **D** is 33 the gain would be $(33/64)^4=0.071$, or -23 dB. The nominal gain setting in the first example would be zero and would be 3 or 4 in the second example.

The overall gain of the chip is given by:

$$G = 2^{\mathbf{GAIN}}(\mathbf{D}/\mathbf{P})^4.$$

In dB this is approximately:

$$G_{dB} = (6x\mathbf{GAIN})+(80 \log(\mathbf{D}/\mathbf{P})).$$

If the gain is too high, the chip will clip the samples to +/- full scale and will set the overflow bit in the output status register (See Section 3.7).

3.6 DECIMATION CONTROL REGISTERS

These registers set the decimation ratio in the programmable low-pass filter.

ADDRESS 8: Decimation Byte 0

BIT	TYPE	NAME	DESCRIPTION
0-7	R/W	**DEC[0:7]**	8 LSBs of **DEC**

ADDRESS 9: Decimation Byte 1

BIT	TYPE	NAME	DESCRIPTION
0-5	R/W	**DEC[8:13]**	6 MSBs of **DEC**
6,7	R/W	-	Unused

A decimation ratio of **D** is set by programming **DEC** to be equal to (**D**-1). The minimum value of **DEC** is 15 and the maximum is 16,383 corresponding to the minimum decimation of 16 and maximum of 16,384.

This document contains information which may be changed at any time without notice

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

3.7 OUTPUT STATUS REGISTER

This register contains flags and status information for the output samples.

ADDRESS 10: Output Status Register

BIT	TYPE	NAME	DESCRIPTION
0	R/W	READY	Tells the chip that the user is ready to capture an output sample. The chip clears this bit when it has captured the sample.
1	R/Clear	MISSED	The chip sets this bit high if a new output sample was ready but the user had not set READY high. This lets the user know if a sample has been missed. This bit is cleared by writing a 0 to the bit. Attempting to write a 1 to this bit does nothing.
2	R/W	OVERFLOW	The chip sets this bit when an overflow occurs. This bit can be used to indicate if the gain is set too high. This bit stays high until the user clears it.
3	R/W	INT_ENABLE	This bit is used in the interrupt mode (See bit 4 below) to turn on the interrupt output. If this bit is off the **$\overline{\text{INT}}$** output pin is forced high. When this bit is high the **$\overline{\text{INT}}$** output pin is equal to READY. When READY goes low, meaning that a new sample has been captured, the **$\overline{\text{INT}}$** pin will go low. If **$\overline{\text{INT}}$** is tied to a processor's interrupt input, then the processor will be interrupted whenever a new sample is ready.
4	R/W	INT_MODE	This bit selects the **$\overline{\text{INT}}$** signal for output on pin **$\overline{\text{SO}}$/$\overline{\text{INT}}$**. Normally **$\overline{\text{SO}}$** is output on the pin.
5	R/W	STROBE_MODE	This bit selects the one-shot sync strobe for output on the **$\overline{\text{SO}}$** pin. When this bit is low the **$\overline{\text{SO}}$** sync is determined by SS_MUX (see Section 3.2).
6	R/W	STROBE	This bit will generate a one clock cycle wide **$\overline{\text{SO}}$** strobe each time it is set. STROBE_MODE must be set before this bit is set and INT_MODE must be low. This bit must be cleared before a new strobe can be generated.
7	R/W	REAL_FS	Used in the real output mode to double the frame rate. If this bit is not set in the real mode, then the chip will operate in the GC1011 real output mode. In the GC1011 mode two real samples are output every frame and the frame rate remains the same as in the complex mode. The I stream will output the even indexed time samples (time=0,2,4,6,...) and the Q stream will output the odd time samples.

The READY signal is used to capture output samples and to read them into an external processor. The user captures outputs by setting the READY bit and then waiting for the bit to be cleared by the chip. When the bit goes low the processor can read the samples out of the I and Q output registers described in

This document contains information which may be changed at any time without notice

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

section 3.9. The processor can wait for READY to go low by either continuously reading this register, or it can use the interrupt output **$\overline{\text{INT}}$** to tell it when the sample is ready. To use the interrupt output mode the user must tie the **$\overline{\text{SO}}$/$\overline{\text{INT}}$** output pin from the chip to an interrupt input of the processor and must set the INT_MODE bit in this register. The processor can then capture samples by setting READY and then setting INT_ENABLE (INT_ENABLE should be set after READY in order to avoid a spurious interrupt due to the interrupt being enabled before READY has settled to its high state). The processor will be interrupted when READY goes low again. When it is interrupted the processor can turn off INT_ENABLE, read the I/Q outputs, and then start over again.

The MISSED flag is provided to let the processor know if it has taken too long to read the I/Q samples before rearming the READY bit. If the processor wants to use the MISSED flag it should clear the flag the first time it sets the READY bit and then check it after setting the READY bit thereafter. The READY bit is set and the MISSED bit cleared by writing a 01(hex) to this register. The READY bit is set and the MISSED bit is left alone by writing a 03(hex) to this register. If interrupts are used, then the user should also set the INT_MODE bit at the same time by writing an 11(hex) or 13(hex) to the register followed by a 1B(hex) to enable the interrupts.

The one-shot strobe is generated by turning INT_MODE off, STROBE_MODE on and STROBE off. **$\overline{\text{SO}}$** strobes are then generated by setting the STROBE bit. This is done by writing a 20(hex) to this register followed by a 60(hex).

This document contains information which may be changed at any time without notice

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

3.8 CHECKSUM REGISTER

This read-only register stores the checksums generated in the diagnostic mode.

ADDRESS 11: CHECKSUM

BIT	TYPE	NAME	DESCRIPTION
0-7	R	CHECK[0:7]	The 8 bit checksum. The checksum is generated as a non-linear feedback accumulation of the **BS**, **FS**, **I**, and **Q** output bits. The current checksum is stored in this register and the checksum generator is cleared whenever the internal sync goes low (see SS_MUX in Section 3.2 for the modes of the internal sync).

3.9 I AND Q OUTPUT REGISTERS

These registers are used to capture output samples.

ADDRESS 12: I-Output Byte 0

BIT	TYPE	NAME	DESCRIPTION
0-7	R	**I[0:7]**	Least significant 8 bits of the **I** output.

ADDRESS 13: I-Output Byte 1

BIT	TYPE	NAME	DESCRIPTION
0-7	R	**I[8:15]**	Most significant 8 bits of the **I** output.

ADDRESS 14: Q-Output Byte 0

BIT	TYPE	NAME	DESCRIPTION
0-7	R	**Q[0:7]**	Least significant 8 bits of the **Q** output.

ADDRESS 15: Q-Output Byte 0

BIT	TYPE	NAME	DESCRIPTION
0-7	R	**QI[8:15]**	Most significant 8 bits of the **Q** output.

The user reads the **I** and **Q** outputs through these read-only registers. The captured samples can be used for gain control, analysis, display, or diagnostics.

This document contains information which may be changed at any time without notice

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

4.0 SPECIFICATIONS

4.1 ABSOLUTE MAXIMUM RATINGS

Referenced to GND:

PARAMETER		MIN	TYP	MAX	UNITS
V_{CC}	DC Supply Voltage	-0.3		7	V
V_{IN}	Input voltage (undershoot and overshoot)	-0.7		V_{CC}+0.7	V
T_{STG}	Storage Temperature	-65		150	^{o}C
F_{CK}	Clock Rate[1]	10			Hz

4.2 RECOMMENDED OPERATING CONDITIONS

PARAMETER		MIN	TYP	MAX	UNITS
V_{CC}	DC Supply Voltage	3.0	5.0	6.0	V
T_A	Temperature Ambient, no air flow	0		60	^{o}C

4.3 DC CHARACTERISTICS

All parameters are at V_{CC}=5V and temperature is 0 to 60 ^{o}C ambient unless noted

PARAMETER			MIN	TYP	MAX	UNITS
V_{IL}	Voltage input low				0.8	V
V_{IH}	Voltage input high	V_{CC}=5V	2.0			V
		V_{CC}=3.3V	1.5			V
I_{IN}	Input current (V_{IN}=0V)			+/-50		µA
V_{OL}	Voltage output low (I_{OL} = 4mA)				0.5	V
V_{OH}	Voltage output high (I_{OH} = -4ma)	V_{CC}=5V	2.5		5	V
		V_{CC}=3.3V	1.5		3.3	V
C_{IN}	Data input capacitance (All inputs except **CK** and **C[0:7]**)			4		pF
C_{CK}	Clock input capacitance (**CK** input)			8		pF
C_{CON}	Control data capacitance (**C[0:7]** I/O pins)			6		pF

1. The chip may be damaged if power is applied for more than 1 second and the clock is below this rate.

This document contains information which may be changed at any time without notice

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

4.4 AC CHARACTERISTICS

Commercial temperature range (0 to 60°C), V_{CC} +/- 5%, unless noted.

PARAMETER					MIN	TYP	MAX	UNITS
F_{CK}	Clock frequency[1]	GC1011A	V_{CC}=	5v	0.1		70	MHz
				3.3v	0.1		42	
		GC1011	V_{CC}=	5v	20		70	MHz
				3.3v	0.1		42	
t_{CKL}	Clock low period (Below V_{IL})		V_{CC}=	5v	6			ns
				3.3V	10			
t_{CKH}	Clock high period (Above V_{IH})		V_{CC}=	5v	6			ns
				3.3v	10			
t_{SU}	Data setup (**X**,$\overline{\textbf{AS}}$, or $\overline{\textbf{SS}}$) before **CK** goes high							
			V_{CC}=	5v	2			ns
				3.3v	4			
t_{HD}	Data hold time (**X**,$\overline{\textbf{AS}}$, or $\overline{\textbf{SS}}$) after **CK** goes high				0			ns
t_{DLY}	Data output delay (**I**, **Q**, **BS**, **FS**, or $\overline{\textbf{SO}}$) from **CK**[2]							
			V_{CC}=	5v			9	ns
				3.3v			14	
t_{CSU}	Control setup before $\overline{\textbf{CS}}$ goes low (**A**, **R/**$\overline{\textbf{W}}$ during read, and **A**, **R/**$\overline{\textbf{W}}$**, C** during write)				5			ns
t_{CHD}	Control hold after $\overline{\textbf{CS}}$ goes high (**A**, **R/**$\overline{\textbf{W}}$ during read, and **A**, **R/**$\overline{\textbf{W}}$**, C** during write)				5			ns
t_{CSPW}	Control strobe pulse width (Write operation)				20			ns
t_{CDLY}	Control output delay to **C** after $\overline{\textbf{CS}}$ goes low[3] (Read operation)						30	ns
t_{CZ}	Control tristate delay after $\overline{\textbf{CS}}$ goes high						10	ns
I_{CCQ}	Quiescent supply current (V_{IN}=0 or V_{CC}, F_{CK}=100Hz)						200	µA
I_{CC}	Supply current (F_{CK}=50 MHz, V_{CC}=5.0V, **D**=64)						150	mA

$$ICC\ (MAX) = \left[\frac{VCC}{5}\right]\left[\frac{FCK}{50'M}\right]\left[0.56 + 0.44\left[\frac{64}{'D}\right]\right] 150\ mA$$

1. The chip may not operate properly at clock frequencies below MIN and above MAX, **D** is the decimation ratio (see Section 3.6)
2. Capacitive output load is 20 pf. Delays are measured from the rising edge of the clock to the output level rising above V_{IH} or falling below V_{IL}.
3. Capacitive output load is 80 pf.

This document contains information which may be changed at any time without notice

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

5.0 APPLICATION NOTES

5.1 POWER AND GROUND CONNECTIONS

The GC1011A chip is a very high performance chip which requires solid power and ground connections to avoid noise on the V_{CC} and GND pins. If possible the GC1011A chip should be mounted on a circuit board with dedicated power and ground planes and with at least two decoupling capacitors (0.01 and 0.1 µf) adjacent to each GC1011A chip. If dedicated power and ground planes are not possible, then the user should place decoupling capacitors adjacent to each V_{CC} and GND pair.

IMPORTANT

The GC1011A chip may not operate properly if these power and ground guidelines are violated.

5.2 STATIC SENSITIVE DEVICE

The GC1011A chip is fabricated in a high performance CMOS process which is sensitive to the high voltage transients caused by static electricity. These parts can be permanently damaged by static electricity and should only be handled in static free environments.

5.3 70 MHZ OPERATION

Care must be taken in generating the clock when operating the GC1011A chip at its full 70 MHz clock rate. The user must insure that the clock is above 2 volts for at least 6 nanoseconds and is below 1 volt for at least 6 nanoseconds. At 70 MHz the clock period is only 14 nanoseconds so that the clock must have a duty cycle of exactly 50%, and the rise and fall times can only be one nanosecond each. One must also be careful to prevent clock undershoot below ground. An ideal clock at 70 MHz would be a square wave with a low voltage of 0.5 volts and a high voltage of 2.5 volts.

5.4 REDUCED VOLTAGE OPERATION

The power consumed by the GC1011A chip can be greatly reduced by operating the chip at the lowest V_{CC} voltage which will meet the application's timing requirements. When operating at a reduced voltage, GRAYCHIP recommends driving the GC1011A chip inputs with the 5 volt to 3 volt interface chips sold by Performance Semiconductor.

This document contains information which may be changed at any time without notice

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

5.5 SYNCHRONIZING MULTIPLE GC1011A CHIPS

A system containing a bank of GC1011A chips will need to be synchronized so that the output frames from each chip are aligned, and, if desired, so that their frequency accumulators are running synchronously. The GC1000 Input Switch chip has built in sync counters which are designed specifically for this purpose. If the GC1000 chip is not used, then the one-shot strobe (see Section 3.7) can be used. The bank of chips should be interconnected so that the **$\overline{\text{SO}}$** pin of one GC1011A chip is tied to the **$\overline{\text{SS}}$** input of all of the chips. The one-shot strobe mode can then be used to simultaneously synchronize all of the chips. The **$\overline{\text{SO}}$** pin of a second GC1011A chip should be tied to the **$\overline{\text{AS}}$** input of all of the chips. The one-shot mode of the second chip can be used to synchronize the frequency accumulators whenever the tuning frequency has been changed.

5.6 PROCESSING COMPLEX DATA

Two GC1011A chips can be used to process complex input data by using one chip to process the I-input data and the other to process the Q-input data. If the two chips are synchronized as discussed above, then the complex output stream can be reconstructed by adding and subtracting the I and Q outputs of the two chips. A programmable gate array chip such as from XILINX would be ideal for this post-processing. The configuration for processing complex data is illustrated in Figure 11.

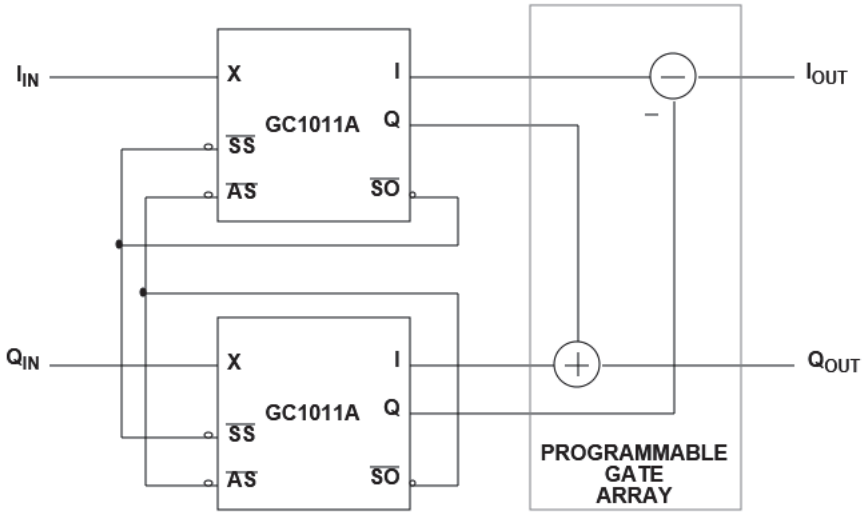


Figure 11. PROCESSING COMPLEX INPUT DATA

This document contains information which may be changed at any time without notice

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

5.7 EXAMPLE RECEIVER ARCHITECTURE

An example digital receiver architecture using the GC1011A chip is shown in Figure 12.

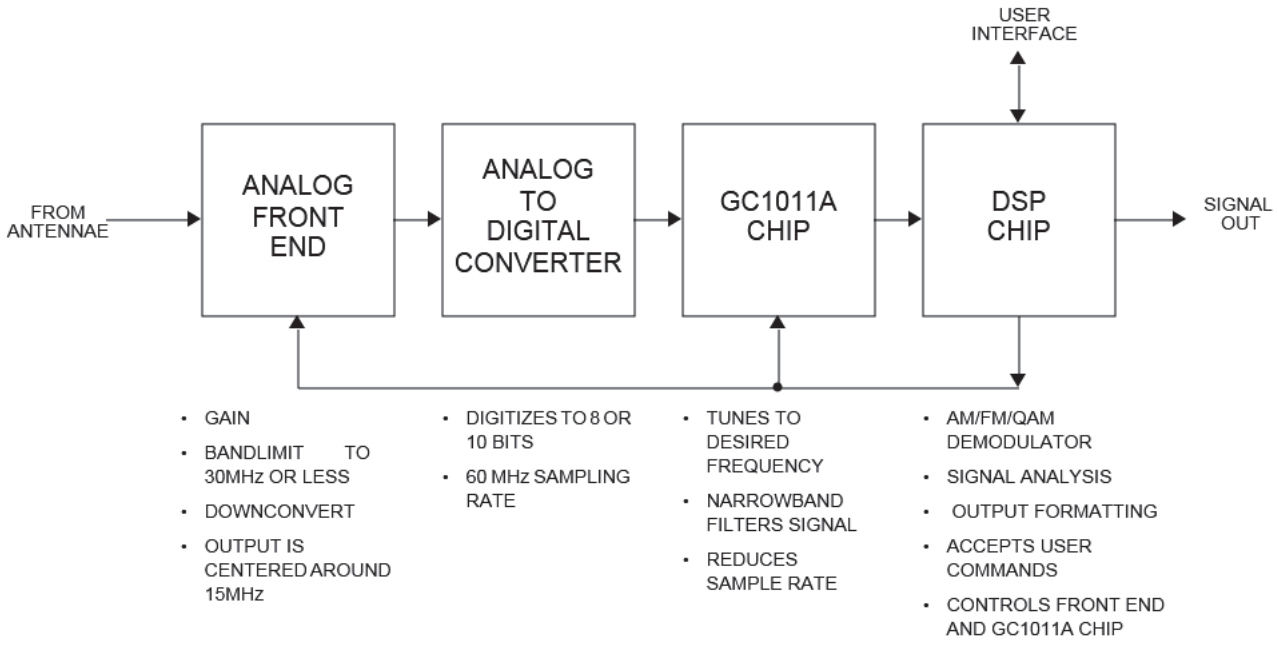


Figure 12. EXAMPLE DIGITAL RECEIVER ARCHITECTURE

The receiver contains an analog front end which downconverts up to 30MHz of radio spectrum to an IF frequency around 15MHz[1]. It also adjusts the gain of the signal so that it fills the dynamic range of the analog to digital converter (ADC). The ADC digitizes the signal to 8 or 10 bits[2] at a sampling rate of 60 MHZ. The GC1011A chip tunes, downconverts, and narrowband filters desired frequencies from within the 30 MHz band. The GC1011A output is then processed by a DSP chip for any final filtering, demodulation or signal analysis. The DSP chip also handles the user interface and provides tuning controls to the GC1011A chip and analog front end.

For applications which do not require the DSP chip at the back end, a simplified architecture can realized by replacing the DSP chip with an digital to analog converter.

1. Note that the HF spectrum (1 to 30MHz) can be digitized directly.
2. The GC1011A chip will accept a 12 bit ADC output as soon as 60 MHZ 12 bit ADCs are available.

This document contains information which may be changed at any time without notice

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Artisan Technology Group is an independent supplier of quality pre-owned equipment

Gold-standard solutions

Extend the life of your critical industrial, commercial, and military systems with our superior service and support.

We buy equipment

Planning to upgrade your current equipment? Have surplus equipment taking up shelf space? We'll give it a new home.

Learn more!

Visit us at **artisantg.com** for more info on price quotes, drivers, technical specifications, manuals, and documentation.

Artisan Scientific Corporation dba Artisan Technology Group is not an affiliate, representative, or authorized distributor for any manufacturer listed herein.

We're here to make your life easier. How can we help you today?

(217) 352-9330 | sales@artisantg.com | artisantg.com

ARTISAN® TECHNOLOGY GROUP