Purpose and Scope

The purpose of this document is to describe the detailed functionality of the second-generation Advanced GPS/GLONASS ASIC, the T7905E (also refered to as the AGGA-2). It describes the functionality, its modes of operation, programming aspects, external interfaces, etc. to a level of detail letting users understand its functions and operation sufficiently for performing the hardware and software design of an T7905E based GNSS receiver.

The AGGA-2 supports a variety of GPS/GLONASS receivers and instruments for space applications, including high precision scientific applications such as atmospheric sounding by radio occultation, Spacecraft Control with navigation and attitude.

Description

The AGGA-2 has 12 single-frequency channels, each capable of tracking a GNSS C/A-code signal on any carrier frequency. Two single-frequency channels can be configured for attitude determination according to the Hybrid Parallel-Multiplex Architecture (RD5).

Three single-frequency channels together with a P-code Unit can be configured into one dual-frequency channel capable of tracking a GNSS C/A-code signal on one carrier frequency (e.g. L1) and a GNSS P-code or Y-code on two carrier frequencies (e.g. L1 and L2). Such configurations can be repeated as well as mixed, as an extreme example it would be possible to have two dual-frequency channels, two channels for attitude determination and two single-frequency channels operating at the same time in one AGGA-2.

The AGGA-2 is highly configurable, allowing a large number of channel settings and slaving schemes. Examples of configuration parameters include: selection of GNSS C/A-code or P-code; setting of code spacing for despreading, etc. Examples of channel slaving include: fast acquisition; multipath identification; Parallel or Hybrid Parallel-Multiplex attitude determination, etc.

The AGGA-2 block diagram is shown in Figure 1. The AGGA-2 contains the following functional blocks:

- -Front-end Interface, shown as eight Real-to-Complex Converters and the Signal Level Detector;
- -Channel Matrix, shown as twelve Channels and four P-code Units;
- -Time Base Generator;
- -Antenna Switch Controller;
- -System Support Functions, with clock generation and Input/Output functionality;
- -Microprocessor Interface, also containing an Interrupt Controller.

It is designed on Atmel M62265E sea of gates matrix and packaged into an MQFPL160.

WIRELESS & µC

Rad Tolerant GPS/GLONASS ASIC

User's Manual

T7905E

Rev. A–24-Aug-01

Definitions and Terms

Channel (sometimes *single-frequency channe*l): The functionality or hardware required to track one pseudo-random code on a single carrier frequency, for example C/A-code on the L1 frequency from Space Vehicle 3, or P-code on the L2 frequency from Space Vehicle 19. All channels of the AGGA-2 do not have identical functionality, as an example some channels are only able to track C/A-code while other channels are also able to track P-code.

Code epoch: The repetition period of a code sequence. In GNSS C/A-code sequences the code epochs are delimited by the “all-ones” state of the code generator shift register. As an example, a GPS C/A-code epoch corresponds to 1023 code chips.

*Complex signa*l: A signal split into in-phase (I) and quadrature (Q) components. *Dual-frequency channe*l: The functionality and hardware needed to track the three ranging signals transmitted by one GPS or GLONASS satellite. These ranging signals are based on two pseudo-random codes (C/A-code and P-code or Y-code) which are both transmitted on the L1 carrier, with the same P-code or Y-code being transmitted on a second carrier L2. In the AGGA-2 a dual-frequency channel is formed by slaving three single-frequency channels.

*Firmwar*e: The low-level software routines directly dealing with the GNSS signal processing hardware, such as closing tracking loops etc.

*GNSS signa*l: Encompassing term for GPS, GLONASS and various augmentation signals including EGNOS, WAAS and MSAS.

*Observable*s: Encompassing term for the carrier frequency and phase for each channel, and the code frequency and phase for each channel. The observables are periodically sampled, and can be used by the receiver firmware to derive the position, time, velocity, and in some cases attitude.

Pseudorange measurement: Synonym for code-phase measurement.

Semi-codeless processing: Processing of the GPS Y-code (encrypted P-code) based on the assumption that the encryption has been performed by a pseudo-random code called W-code, having a chip period that is an integer multiple of the P-code chip period.

Bit Numbering and Naming Conventions

Buses and vectors are indexed with the Least Significant Bit (LSB) having the index 0 and the Most Significant Bit (MSB) having index N-1 for a bus of width N bits.

The stages in a Linear Feedback Shift Register (LFSR) are numbered so that stage one receives the feedback term and the shifts are performed in the direction from lower to higher stage number. An LFSR tap has the same number as the stage it is output from.

When a vector value is shown, the left-most bit is the one with the highest number. This also applies for LFSR vectors. Active low signals are indicated by being overlined, as in *Reset*.

FUNCTIONAL OVERVIEW

This section gives an overview of the AGGA-2 functionality. Detailed descriptions can be found in subsequent sections.

Front-end Interface

In Figure 1 the Front-end Interface is shown as eight Real-to-Complex Converters and the Signal Level Detector. The AGGA-2 has eight separate 2-bit inputs *ADCIn0* to *ADCIn*7, supporting four different input formats: sign/magnitude, unsigned, two's complement and comparator ladder.

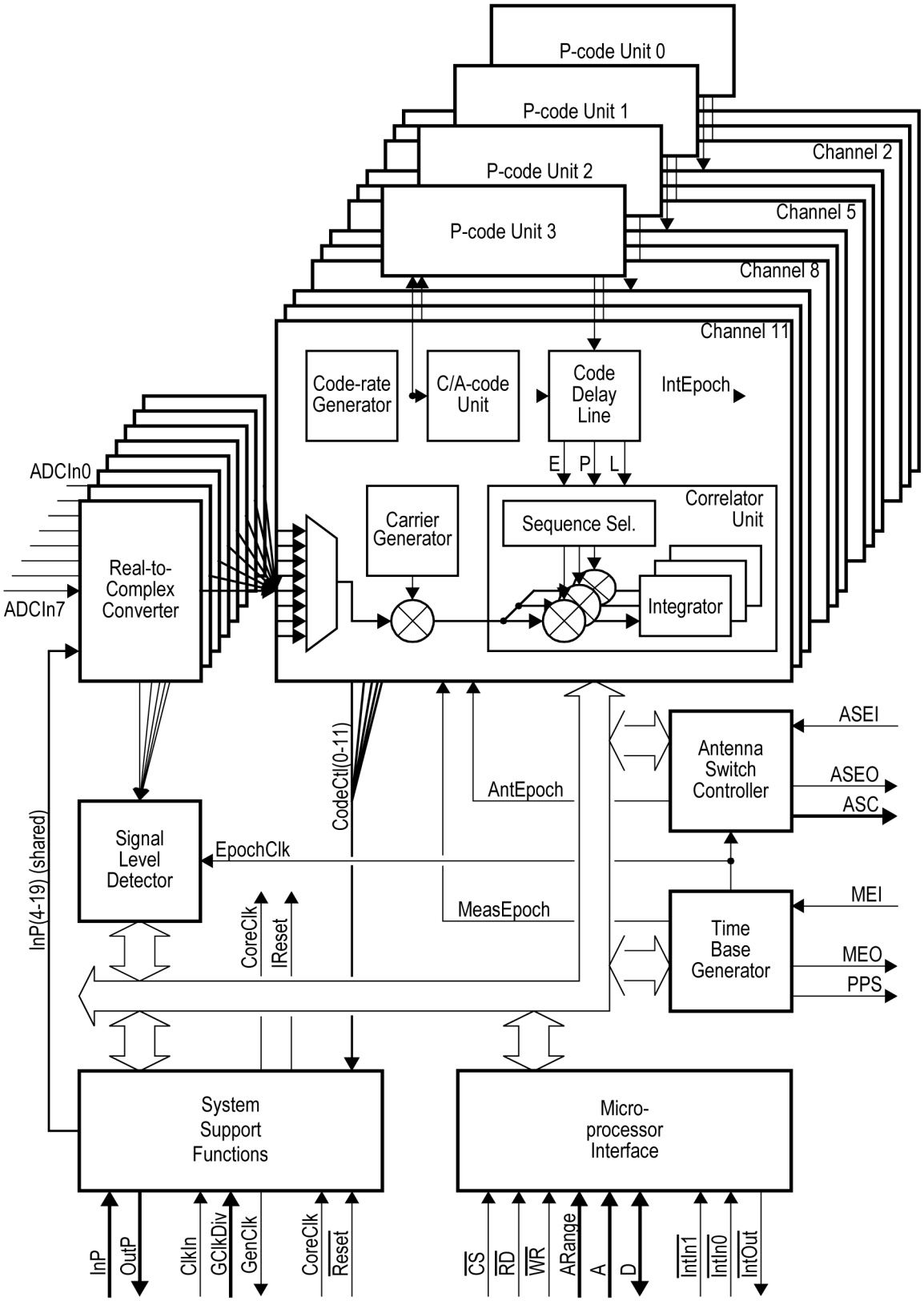
Each of the eight inputs can accept a digitised Intermediate Frequency (IF) input signal in real format, centred around one quarter of the sampling frequency. In this case, the input signal is transformed by the associated Real-to-Complex Converter into an in-phase (I) component and a quadrature (Q) component, and at the same time it is down-converted by a fixed frequency equal to one quarter of the sampling frequency.

Alternatively, the AGGA-2 can accept an input signal already having complex format represented as 2-bit I and Q values on two adjacent *ADCIn* inputs. The Real-to-Complex Converter is then bypassed and no down-conversion is performed. To support redundancy, four additional complex inputs in sign/magnitude format can be accessed by sharing 16 of the *InP* pins with the System Support Functions.

The Signal Level Detector allows to measure an estimate of the input signal plus noise energy to be used for Automatic Gain Control (AGC) purposes. It integrates an estimate of the input power from one of the I or Q branches at a time, and it is time shared by all inputs.

The Front-end Interface is described in page 8.

Figure 1. AGGA-2 block diagram



Channel Matrix

In Figure 1 on page 4 the Channel Matrix is shown as twelve channels together with four P-code Units. Although not indicated in the figure, each third channel is a CA-channel which can operate with C/A-code only, while the remaining ones are CaP-channels which are able to operate with either C/A-code, P-code or Y-code. One CA-channel together with two CaP-channels and one P-code Unit can be configured as a dual-frequency channel capable of tracking C/A-code on one channel and P-code or Y-code (using semi-codeless processing) on two channels. Channels can also be slaved for attitude determination, fast acquisition and multipath identification.

Each channel contains an Input Selector; the Carrier Generator and the Final Down-converter for down-conversion to baseband; the Code-rate Generator and the C/A-code Unit for generating the replica GNSS C/A-codes; the Code Delay Line for code selection and spacing; and the Correlator Unit with three despreaders and complex integrators. The CA-code or P-code of any channel can be output by the Code & Control *(CodeCt*l) signals from each channel via the System Support Functions.

Each P-code Unit generates one replica P-code sequence, a delayed version of this P-code sequence, and the control strobes required for dual-frequency operation including semi-codeless processing.

At the end of each Integration Epoch *(IntEpoc*h, individual for each channel) the correlation values for that channel are latched into the correlation registers. These correlation values can be used by the receiver firmware to close the tracking loops, including carrier and code phase error computation, application of loop filters, estimation of signal and noise amplitude, etc. At the end of each Measurement Epoch *(MeasEpoc*h) the carrier and code phase values for all channels are sampled into the observables registers. These observables can be used by the receiver firmware to derive the position, time, velocity and attitude.

The Channel Matrix is described on page 13.

Time Base Generator

The Time Base Generator produces the *EpochClk* clock, the Measurement Epoch *(MeasEpoc*h) strobe and the Pulse-Per-Second *(PP*S) strobe. *EpochClk* has a relatively low frequency and is used by the Signal Level Detector and the Antenna Switch Controller. *MeasEpoch* signals the end of each Measurement Epoch and is used in the Channel Matrix to sample the observables. It is taken from the Measurement Epoch Input *(ME*I), which is externally connected to the Measurement Epoch Output *(ME*O) from the same AGGA-2, or in receivers with multiple AGGA-2s from one master AGGA-2. *MEO* is generated from *EpochCl*k. The *PPS* output is intended for synchronising external equipment to the receiver time, and it is derived from *EpochCl*k.

The Time Base Generator is described in page 41.

Antenna Switch Controller

The Antenna Switch Controller supports the Hybrid Parallel-Multiplex attitude determination scheme (RD5) through the Antenna Switch Control *(AS*C) outputs intended for controlling external antenna switching of up to four antennas. It also generates the Antenna Switch Epoch *(AntEpoc*h) strobe which signals antenna switch events, and which is used to control the updating of the correlation values for the slaved channels. Both the *AntEpoch* strobe and the *ASC* outputs are generated from the Antenna Switch Epoch Input *(ASE*I) which is externally connected to the Antenna Switch Epoch Output *(ASE*O) from the same AGGA-2, or in receivers with multiple AGGA-2s from one master AGGA-2. *ASEO* is generated from the *EpochClk* clock.

The Antenna Switch Controller is described in page 44.

System Support Functions

The System Support Functions include the generation of an external clock, reset handling, a basic parallel Input/Output interface and GNSS code output.

The clock generator produces the General Purpose Clock *(GenCl*k). This independent-clock is derived from the *ClkIn* clock input and can be used for clocking the *CoreClk* input, the GP2010 front-end (RD10), the receiver micro-processor, etc. The clock frequency is determined by the 3-bit *GClkDiv* input. The AGGA-2 is reset when the *Reset* input is asserted or by a microprocessor write access to a specific register. The *CoreClk* clock input provides the AGGA-2 internal clock.

The parallel Input/Output interface consists of the 20-bit Input Port *(In*P) and the 12-bit Output Port *(Out*P). They can be used for functions such as monitoring lock indicators of RF down-converters and power-down of external circuitry. Two of the outputs can be configured to provide the code sequence and the corresponding Integration Epoch of any channel, which can be used for GNSS transmission, calibration or debugging. The System Support Functions are described in page 46.

Microprocessor Interface

A microprocessor can access the AGGA-2 registers through the microprocessor interface as a generic memory-mapped peripheral. The interface consists of the 10-bit address bus *(*A), the 32-bit data bus *(*D), and the Chip Select *(C*S), Read *(R*D) and Write *(W*R) strobes. It is compatible with both the ADSP21020 DSP microprocessor (RD1) and the ERC32 Sparc chip set (RD7, RD8, RD9). By hardwiring the 2-bit *ARange* input, up to four different AGGA-2s can be accessed using a single chip select signal.

As correlation values, observables etc. become available in the AGGA-2, an external interrupt can be generated to inform the microprocessor that these can be read out. The Interrupt Controller stores the interrupt status and generates the external interrupt request output *IntOu*t. The AGGA-2 has two external interrupt inputs *IntIn0* and *IntIn*1, allowing multiple AGGA-2s to be connected in a tree configuration to a single interrupt input at the microprocessor. The Microprocessor Interface is described in page 46.

Internal Clocks and Epoch Strobes

This section gives an overview of the internal clocks and epoch strobes in the AGGA-2. *CoreCl*k: This input clock is used by all the blocks in the AGGA-2. *EpochCl*k: The Epoch Clock is generated in the Time Base Generator. It can have a frequency up to 6 MHz, with a typical value being 1 kHz. Its frequency and, indirectly, phase can be accurately controlled allowing synchronisation with a high degree of precision for the strobes derived from this clock. It is used to generate the externally provided Measurement Epoch Output *(ME*O), the Antenna Switch Epoch Output *(ASE*O) and the Pulse-Per-Second *(PP*S) strobes. It is also used to generate the integration period of the Signal Level Detector in the Front-end Interface.

*IntEpoc*h: The Integration Epoch strobes are independently generated for each channel, and indicate the end of each integration period. They are used in the Carrier Generator, the Code-rate Generator, the P-code Unit and the Correlator Unit to update the frequency and phase of the code and carrier replica signals, as well as for updating the correlation registers. Each Integration Epoch has a period in a range from 1/4th of a C/A-code epoch to 20 C/A-code epochs. Typically, the period is 1/4 to 1 C/A-code epoch during acquisition and 10 or 20 C/A-code epochs when tracking a GNSS signal.

*MeasEpoc*h: The Measurement Epoch strobe is taken from the Measurement Epoch Input*(ME*I) and is used to periodically sample the observables. *MeasEpoch* is common for all channels and P-code Units in a receiver.

*AntEpoc*h: The Antenna Switch Epoch strobe is taken from the Antenna Switch EpochInput *(ASE*I) and is used to control the antenna selection and to control the updat-

ing of the correlation values in synchronisation with the antenna switching. It is intended to be used for attitude determination according to the Hybrid Parallel-Multiplex scheme.

Summary of Operation

The AGGA-2 performs the high-speed GNSS signal processing in hardware, while the configuration as well as the control loops, e.g. for acquisition and tracking, are performed by a microprocessor executing dedicated firmware. Without the microprocessor interaction the AGGA-2 will not perform any useful signal processing. This gives the GNSS receiver designer freedom to adapt the functionality and performance for the particular application at hand, but it also means that the results are much dependent on the firmware design. This section will give an overview of typical hardware operation, without going into firmware issues such as acquisition, tracking, etc.

At reset the AGGA-2 is inactive. After programming the appropriate configuration (input signal format, input selection, channel enabling and slaving, correlation mode, interrupt enabling, etc.) and initial parameter settings (carrier and code-rate frequencies, code selection, Integration Epoch length, Time Base settings, etc.), some of these will become effective immediately, whereas others will become effective at the next Integration Epoch or *EpochClk* cycle. In order to allow reprogramming of the latter, all Code-rate Generators and the Epoch Clock Divider are running at certain frequencies after reset, so that Integration Epochs and *EpochClk* cycles occur regularly.

At this point in time, the AGGA-2 will process the input signals according to the parameter settings:

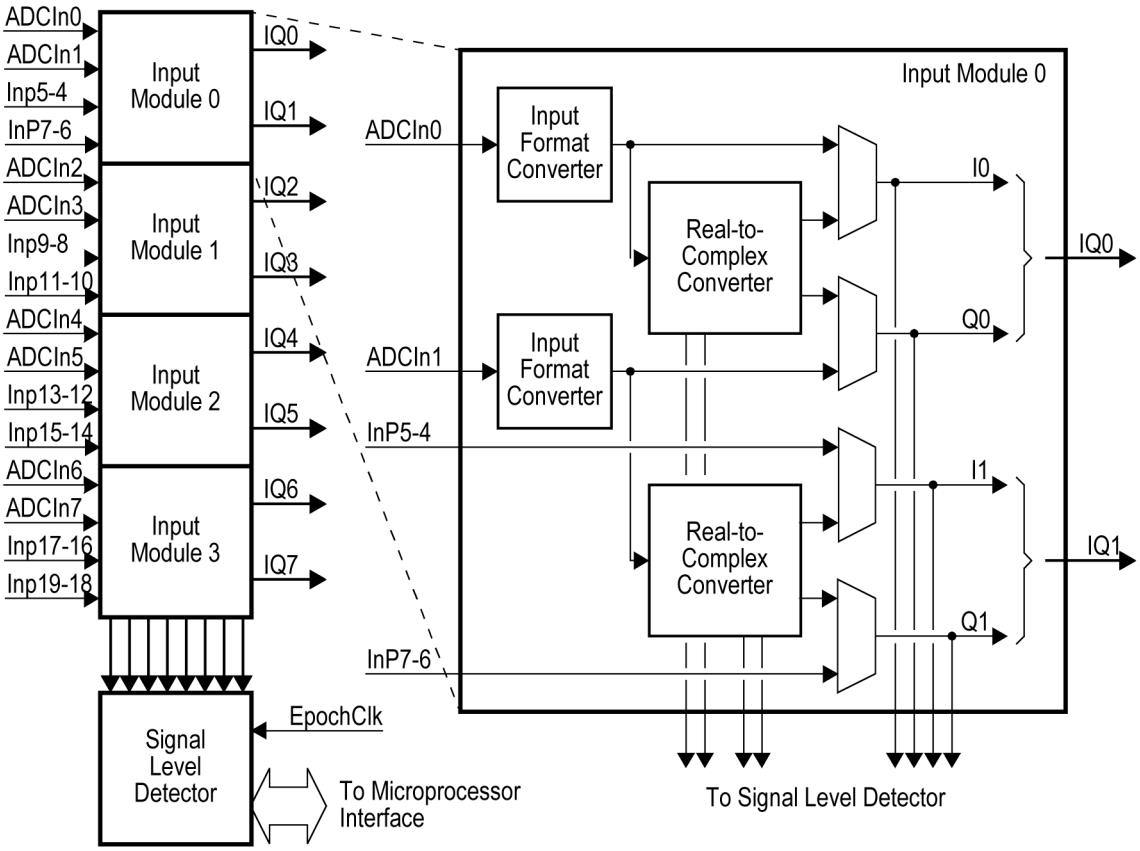
- If so configured, real input signals will be converted into complex signals in the Front-end Interface.
- In enabled channels the Code-rate Generator will provide a chip-rate signal according to the programmed code-rate settings, and the selected code sequence will be generated at this code rate.
- In enabled channels the selected complex input signal will be down-converted according to the carrier settings, despread with the generated code sequence with two or three slightly different phases (as an example: Early, Punctual and Late) and the results integrated over the programmed Integration Epoch into the correlation values.
- At the end of each Integration Epoch (separate for each channel), an interrupt can be generated, allowing the corresponding correlation values to be read out by the microprocessor. Depending on the channel state (e.g. acquisition or tracking) the firmware will calculate new parameters for the carrier frequency and phase, and the code-rate frequency and phase. Occasionally, the channel state will be changed, with associated reprogramming of the correlation mode, Integration Epoch length, and sometimes even complete reprogramming of the channel.
- When several channels are reliably tracking GNSS signals, the current values of the carrier and code phase for all channels can be sampled at points in time called Measurement Epochs. These phase values are referred to as the observables, and constitutes the main input to the calculation of time, position, velocity, etc.

In parallel with the channel operation, the Signal Level Detector can be used to estimate the input signal level, the Time Base generator can be adjusted to provide a reference time strobe, and the Antenna Switch Controller can provide the strobes used for Hybrid Parallel-Multiplex attitude determination, etc.

FRONT-END INTERFACE

This section describes the Front-end Interface. As shown in Figure 2 it consists of eight Real-to-Complex Converters grouped two by two into four Input Modules, and one Signal Level Detector. One of the Input Modules is shown in greater detail.

Figure 2. Front-end Interface block diagram



The AGGA-2 has eight separate 2-bit inputs *ADCIn0* to *ADCIn*7, supporting four different input formats: sign/magnitude, unsigned, two's complement and comparator ladder. The Input Format Converter translates the format used into the AGGA-2 internal 2-bit representation corresponding to the values [-3, -1, +1, +3].

Each of the eight *ADCIn* inputs can accept a digitised Intermediate Frequency (IF) input signal in real format. The signal is transformed by the associated Real-to-Complex Converter into a complex signal and at the same time it is down-converted by one quarter of the sampling frequency. Alternatively, the AGGA-2 can accept a complex signal on two adjacent *ADCIn* inputs, in which case the corresponding Real-to-Complex Converters are bypassed (no down-conversion is then performed). Up to eight real signals or four complex signals can be connected to each AGGA-2 through the *ADCIn* inputs. To support redundancy, four additional complex inputs in sign/magnitude format can be accessed by sharing 16 of the *InP* pins with the System Support Functions. In this case the odd numbered *InP* signal is the sign bit.

For real input signals, the signal should be centred around half the *CoreClk* frequency, with the useful single-sided signal bandwidth being within 0.08·*CoreClk* to 0.92·*CoreCl*k. For complex input signals, the full double-sided bandwidth ±*CoreCl*k/2 can be utilised.

The Front-end Interface provides eight complex outputs *IQ0* to *IQ7* to the Channel Matrix, each having a 2-bit I component and a 2-bit Q component. When complex signals are input the odd *IQ* signals (i.e. *IQ*1, *IQ*3, *IQ5* and *IQ*7) originate from the *InP* inputs.

The Signal Level Detector allows to measure an estimate of the input signal plus noise energy to be used for Automatic Gain Control (AGC) purposes. It integrates an estimate of the input power from one of the I or Q branches at a time, and it is time shared by all Front-end Interface Inputs. The integration interval is programmable between 1 and 255 *EpochClk* periods.

Input Format Converter

The Input Format Converter samples data from the *ADCIn* inputs and translates the format used into the AGGA-2 internal representation.

The *ADCIn* inputs are sampled by the *CoreClk* clock. When the input is in real format, both the rising and falling edges of *CoreClk* are used since two samples are needed per *CoreClk* cycle. When the input is in complex format only the rising edge of *CoreClk* is used. The System Support Functions *GenClk* divider can be used to divide the sampling clock driving the ADCs by a factor of two to obtain the *CoreClk* frequency (see page 46).

The Front-end Interface supports four different input formats: sign/magnitude, unsigned, two's complement and comparator ladder, as defined in Table 1. In the table, the left-most bit corresponds to bit 1 of the respective *ADCIn* inputs. The Input Format Converter translates the format used into the internal representation corresponding to the values [-3, -1, +1, +3] as shown in the *Value* column. The input format is selected by programming the *InputFormat* field of the *InputMode* register.

Table 1. Supported Input Formats

Value	Sign/magnitude	Unsigned	Two's complement	Comparator ladder
-3	01	00	10	00
-1	00	01	11	01
+1	10	10	00	11
+3	11	11	01	10

Note: For the comparator ladder format a four-level comparator ladder should be used, externally XOR-ing the two comparator outputs corresponding to the -3 and +3 values so all four values can be represented. The XOR-ed signal should then be fed to bit 0 of the *ADCIn* input while the output from the middle (zero crossing) comparator should be fed to bit 1.

Note: If the signal provided to the *ADCIn* input only has one bit, to obtain better performance the -3 and +3 values should be used rather than the -1 and +1 values.

Real-to-Complex Converter

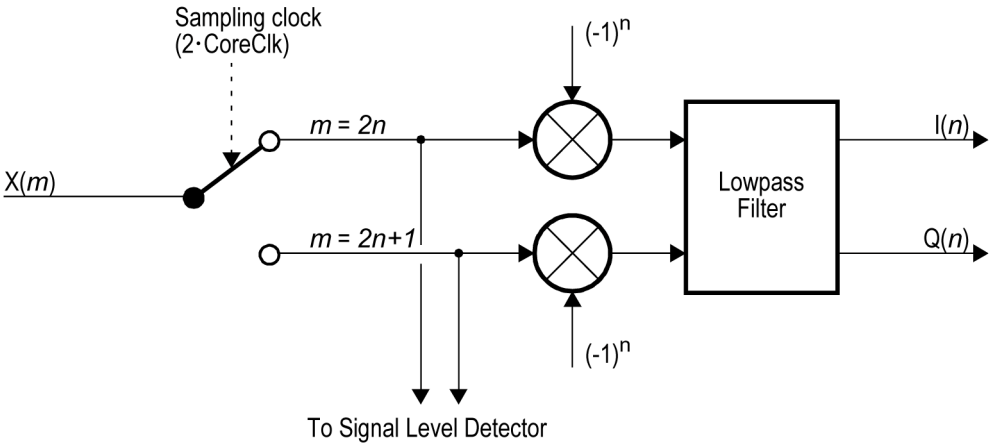
The Front-end Interface supports digitised Intermediate Frequency (IF) input signals in either complex or real format. The input conversion is programmed by the *ConvMode* fields of the *InputMode* register. Each input can also be disabled using the same field. When inputs are disabled or used for complex data, the corresponding Real-to-Complex Converters are automatically disabled in order to reduce the power consumption.

When the input IF signal is in complex format, it can be connected to two adjacent *ADCIn* inputs as a 2-bit I value and a 2-bit Q value. Even-numbered *ADCIn* inputs are

used for the I component while odd-numbered *ADCIn* inputs are used for the Q component. The associated Real-to-Complex Converter is then bypassed, and no down-conversion is performed.

Alternatively, each of the eight *ADCIn* inputs can accept a signal in real format. The input signal is then transformed by the associated Real-to-Complex Converter into an in-phase (I) component and a quadrature (Q) component. This is achieved by decimating the input stream into two streams of even and odd samples at half the input sampling rate. The even and odd data streams are each multiplied by a $(-1)^n$ sequence, with *n* being the sample number on each respective branch. This corresponds to a down-conversion of the input signal by a fixed frequency equal to one quarter of the sampling frequency (i.e. by half the *CoreClk* frequency). The two data streams are subsequently time-aligned using Finite Impulse Response (FIR) lowpass filters. A block diagram of the Real-to-Complex Converter is shown in Figure 3.

Figure 3. Real-to-Complex Converter block diagram



The Lowpass Filter characteristics related to the down-converted complex signal are:

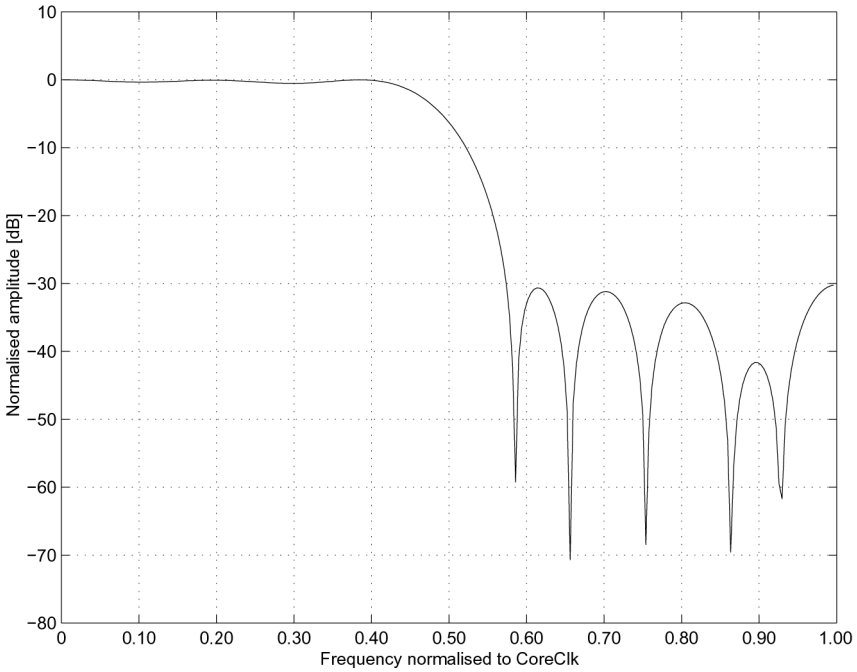
- Passband ripple peak-to-peak less than 0.6 dB;
- Stopband suppression of 30 dB;
- The passband ends at ±0.42 times the *CoreClk* frequency;
- The stopband starts at ±0.58 times the *CoreClk* frequency.

TOS-ESM/PS/100 Issue 1 agency For best results the incoming real signal frequency should be centred around half the *CoreClk* frequency and should have a bandwidth not exceeding 0.84 times the *CoreClk* frequency. A real input signal between 0.08·*CoreClk* and 0.92·*CoreClk* will result in an internal complex signal between -0.42·*CoreClk* and 0.42·*CoreClk*

Note: In the conversion process the input frequency spectrum is inverted. For example, positive frequency offset with respect to *CoreClk*/2 at the *ADCIn* input will appear as a negative frequency in the converted complex signal.

The FIR filter is symmetrical with the following coefficients related to the input sampling rate: [2, 0, -3, 0, 5, 0, -9, 0, 30, 47, 30, 0, -9, 0, 5, 0, -3, 0, 2]. The overall filter response related to the down-converted signal is shown in Figure 4.

Figure 4. Overall Filter response for the Lowpass Filter



The filter output is quantised into the internal two-bit format by hardlimiting the filter output value. A filter output value <-95 corresponds to -3; -95 ϒ⇐filter output value ϒ⇐0 corresponds to -1; 1 ϒ⇐filter output value ϒ⇐95 corresponds to +1, and higher filter output values correspond to +3.

Front-end Interface Outputs

The Front-end Interface provides eight complex outputs *IQ0* to *IQ7* to the ChannelMatrix, each having a 2-bit I component and a 2-bit Q component corresponding to the values [-3, -1, +1, +3]. A complex sample is provided each *CoreClk* period. When complex signals are input the odd *IQ* signals (i.e. *IQ*1, *IQ*3, *IQ5* and *IQ*7) originate from the *InP* inputs. When an input is disabled the corresponding output has the value -1 for both the I and the Q component.

Signal Level Detector

The Signal Level Detector allows to measure an estimate of the input signal plus noise energy which can be used as an indication of the input signal strength. Since the input essentially is a non-coherent noise signal it is possible to separately measure the energy for the I and the Q components. This estimate can be used by the microprocessor to control an AGC loop.

The Signal Level Detector is time shared by all Front-end Interface Inputs. For a complex signal either the I or Q component as provided at the *ADCIn* or *InP* inputs can be selected. For a real signal, it is the unfiltered I or Q component after the decimating switch but before conversion and filtering, as shown in Figure 3. The selection is controlled by the *SignalSel* and *IQSel* fields of the *LevelIntCtl* register. The Signal Level Detector performs the integration by counting either the number of positive samples or the number of samples with maximum magnitude, as programmed by the *SignMagnSel* field of the *LevelIntCtl* register.

The integration interval is determined by a timer which also can generate an interrupt at the end of each integration interval. The length of the integration interval is programmable between 1 and 255 multiples of the *EpochClk* period by the *IntLength* field of the

LevelIntCtl register. The Signal Level Detector is disabled when the integration interval is programmed to 0. The value integrated by the Signal Level Detector is provided in the *LevelVal* field of the *LevelValue* register. The maximum value that can be represented by this 23-bit field corresponds to 279 ms of accumulated data at a *CoreClk* rate of 30 MHz and an *EpochClk* frequency of 1 kHz. The level value counter is reset to zero at the start of each new integration interval. When the Signal Level Detector is disabled the contents of the *LevelValue* register are undefined.

The *LevelIntCtl* register is double-buffered and a newly programmed value becomes effective at the start of the next integration interval. This allows the ongoing integration to complete before the next integration with new parameters starts. When the Signal Level Detector is disabled *(IntLength* field is zero), a newly programmed value becomes effective immediately. The values of the *SignalSe*l, *IQSel* and *SignMagnSel* fields used during the integration are also provided in the *LevelValue* register to allow the firmware to identify the settings corresponding to each generated input energy estimate.

Note: The enabling and disabling of the Signal Level Detector is immediate and asynchronous to the *EpochClk* clock. The duration of the first and the last integration intervals can therefore be up to one *EpochClk* period shorter than the programmed duration.

CHANNEL MATRIX

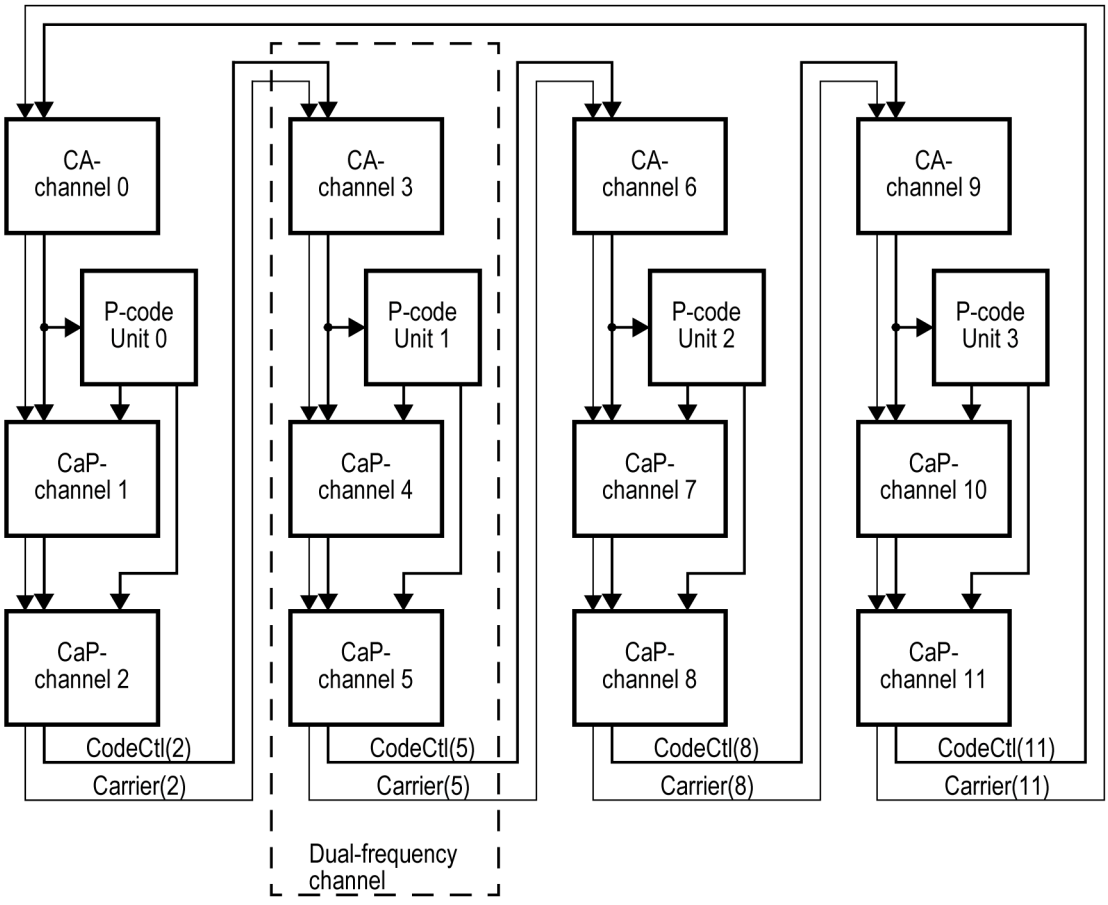
This section describes the Channel Matrix.

Overview

The structure of the Channel Matrix is shown in Figure 5 with the twelve channels and the four P-code Units. As shown in the figure, four channels are CA-channels which can operate with C/A-code only. The other eight channels are CaP-channels which are able to operate with either C/A-code, P-code or Y-code. When processing C/A-code a CaP-channel behaves identically to a CA-channel. Any channel can also be configured to process un-modulated input signals, which could be used for calibration of RF front-end group delay variations.

One CA-channel together with two CaP-channels and one P-code Unit can be configured as a dual-frequency channel capable of tracking C/A-code on one channel and P-code or Y-code on two channels. This is shown as one column of the Channel Matrix in Figure 5. Each P-code Unit generates one replica P-code sequence, a delayed version of this P-code sequence, and the control strobes required for dual-frequency operation including semi-codeless processing.

Figure 5. Channel Matrix with slaving connections for the AGGA-2



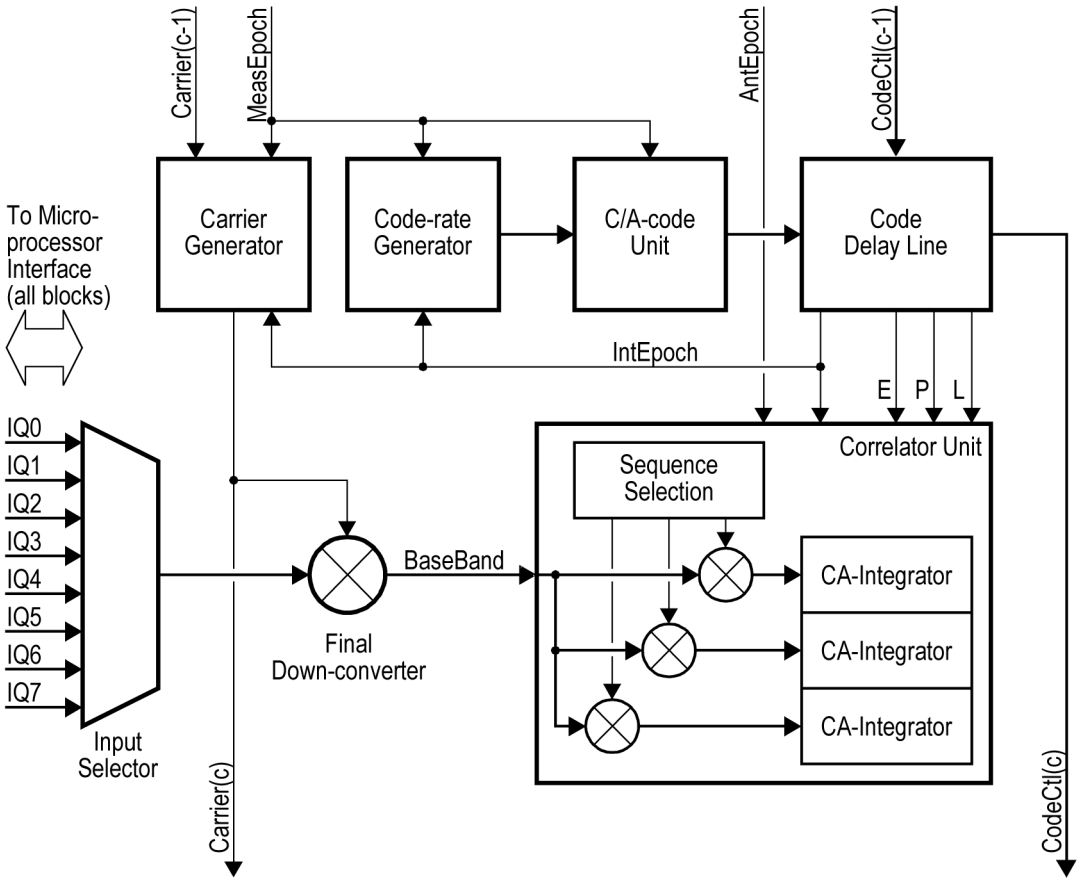
In addition to slaving for dual-frequency operation the channels can also be slaved for attitude determination, fast acquisition and multipath identification.

The Channel Matrix operation is configured by an external microprocessor. The basic functions of each channel are: selection of one of eight complex signals from the Front-end Interface, final down-conversion of the selected complex sign-magnitude signals to baseband, despreading of the complex baseband signal with the on-chip generated replica C/A- or P-code, and integration over a programmable time interval. The result of these integrations is the correlation values. These together with the observables, being the carrier and code frequency and phase measurements, are the outputs from the channel.

CA-channel

A CA-channel can process a C/A-code modulated GNSS signal, but not a P-code or Y-code modulated signal. The CA-channel block diagram is shown in Figure 6.

Figure 6. CA-channel block diagram



As shown in the figure each CA-channel contains an Input Selector, a Carrier Generator, a Final Down-converter; a Code-rate Generator, a C/A-code Unit, a Code Delay Line and a Correlator Unit with three despreaders and complex integrators. Channel slaving and the associated signals *Carrier* and *CodeCtl* are described in page 16.

The Input Selector, described in page 20, selects one of the eight *IQ*7 *0* signals from the Front-end Interface, each providing a complex sample every *CoreClk* cycle. In the Final Down-converter any residual carrier frequency is then removed by rotating the selected

complex signal by an angle determined by the Carrier Generator, resulting in a complex baseband signal. Given an adequate *CoreClk* frequency, the range of the down-conversion is sufficient to allow the removal of the GLONASS channel frequency. The carrier frequency is programmable, and a new value becomes effective at the start of the next Integration Epoch. At the end of each Measurement Epoch the carrier observable is latched into a register to be read by the microprocessor. The Carrier Generator is described in page 21 and the Final Down-converter in page 22.

The Code-rate Generator, described in page 23, determines the chip rate of the C/A-code Unit. The chip rate is programmable, and a new value becomes effective at the start of the next Integration Epoch. The C/A-code Unit generates the replica C/A-code sequence and associated control signals. The GNSS C/A-code generated and the length of the integration period is programmable. At the end of each Measurement Epoch the code observable is sampled so it can be read by the microprocessor. The C/A-code Unit is described in page 24.

In the Code Delay Line the Early *(*E), Punctual *(*P) and Late *(*L) replica code sequences are derived from the single code sequence generated by the C/A-code unit. The chip spacing between the *E* and the P, and between the *P* and the L, replica code sequences is individually programmable. The Integration Epoch *(IntEpoc*h) is also provided from the Code Delay Line, either taken from the C/A-code Unit code control signals or from a slaved code control signal. The Code Delay Line is described in page 26.

The Correlator Unit despreads the complex baseband signal using a combination of three code sequences, with a choice of seven different combination possibilities. The results are integrated during one Integration Epoch. When the CA-channel is configured for Hybrid Parallel-Multiplex attitude determination, the integrator operation is also controlled by the Antenna Switch Epoch. The Correlator Unit has three complex CA-Integrators, each with one accumulator for the I component and one accumulator for the Q component. The output of the Correlator Unit is the correlation values. The Correlator Unit is described in page 31.

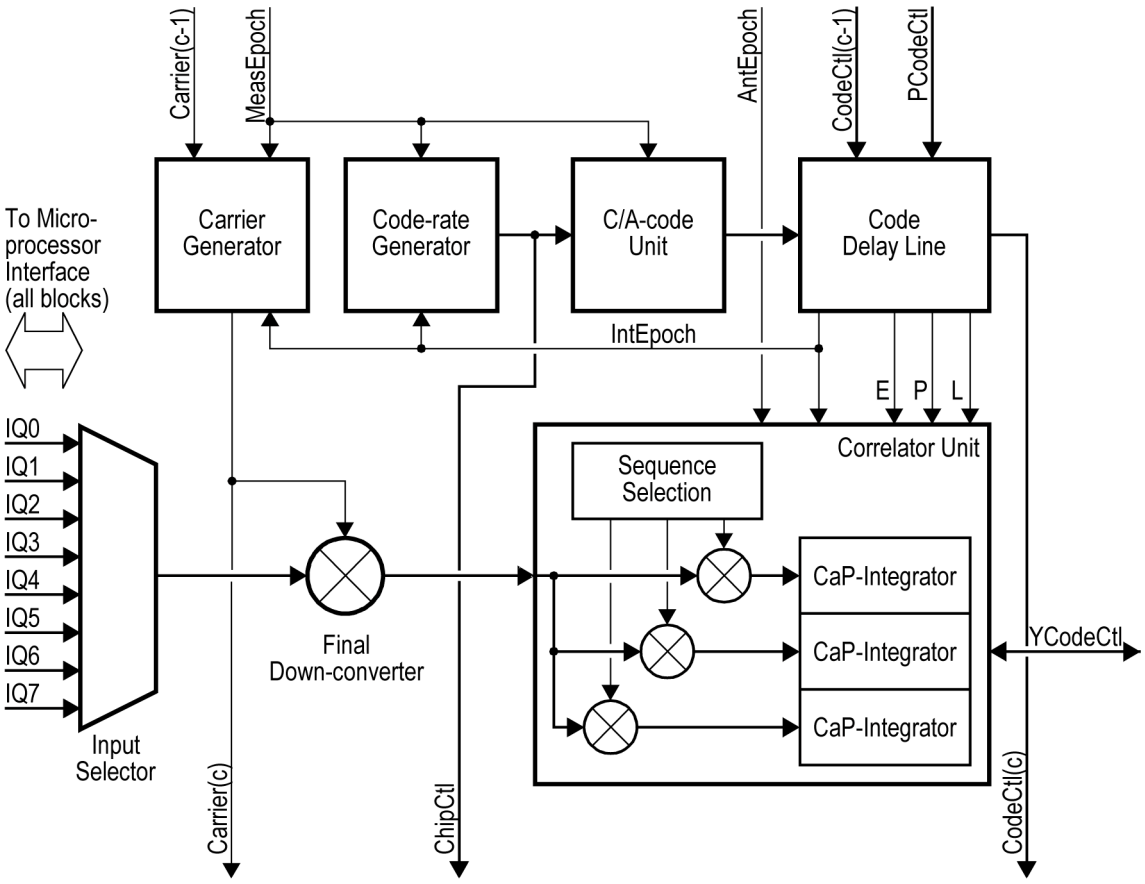
Although not explicitly shown in Figure 6, the Microprocessor Interface is connected to all blocks except the Final Down-converter.

CaP-channel

A CaP-channel can process C/A-code, P-code and Y-code modulated GNSS signals. The

CaP-channel block diagram is shown in Figure 7 space agency 21 TOS-When processing C/A-code a CaP-channel behaves identically to a CA-channel, which is described in page 14. For P-code and Y-code processing a CaP-channel uses the P-code and semi-codeless capabilities of the CaP-Integrators. In addition to a CA-channel, a CaP-channel provides the Chip Control *(ChipCt*l) signal for controlling the chip rate of the associated P-code Unit and can select the P-code & Control *(PCodeCt*l) slaving signal from the same P-code Unit. Furthermore, for semi-codeless operation the CaP-Integrators can be controlled by the bi-directional Y-code Control *(YCodeCt*l) signal provided by the Semi-codeless Controller in the P-code Unit.

Figure 7. CaP-channel block diagram



The CaP-Integrator is described in page 33. Channel slaving for dual-frequency operation is described in page 19 and the adaptive semi-codeless dual-frequency operation for encrypted P-code is described in the following section.

Channel Slaving

Channel slaving is used to expand the basic capabilities of a single channel. Furthermore, being performed in hardware it reduces the microprocessor load compared to firmware slaving. Since disabled units can be powered down it also reduces the power consumption. Up to five single-frequency channels can be slaved any master channel at worst case operating conditions. It is possible to configure multiple different groups of slaved channels at the same time.

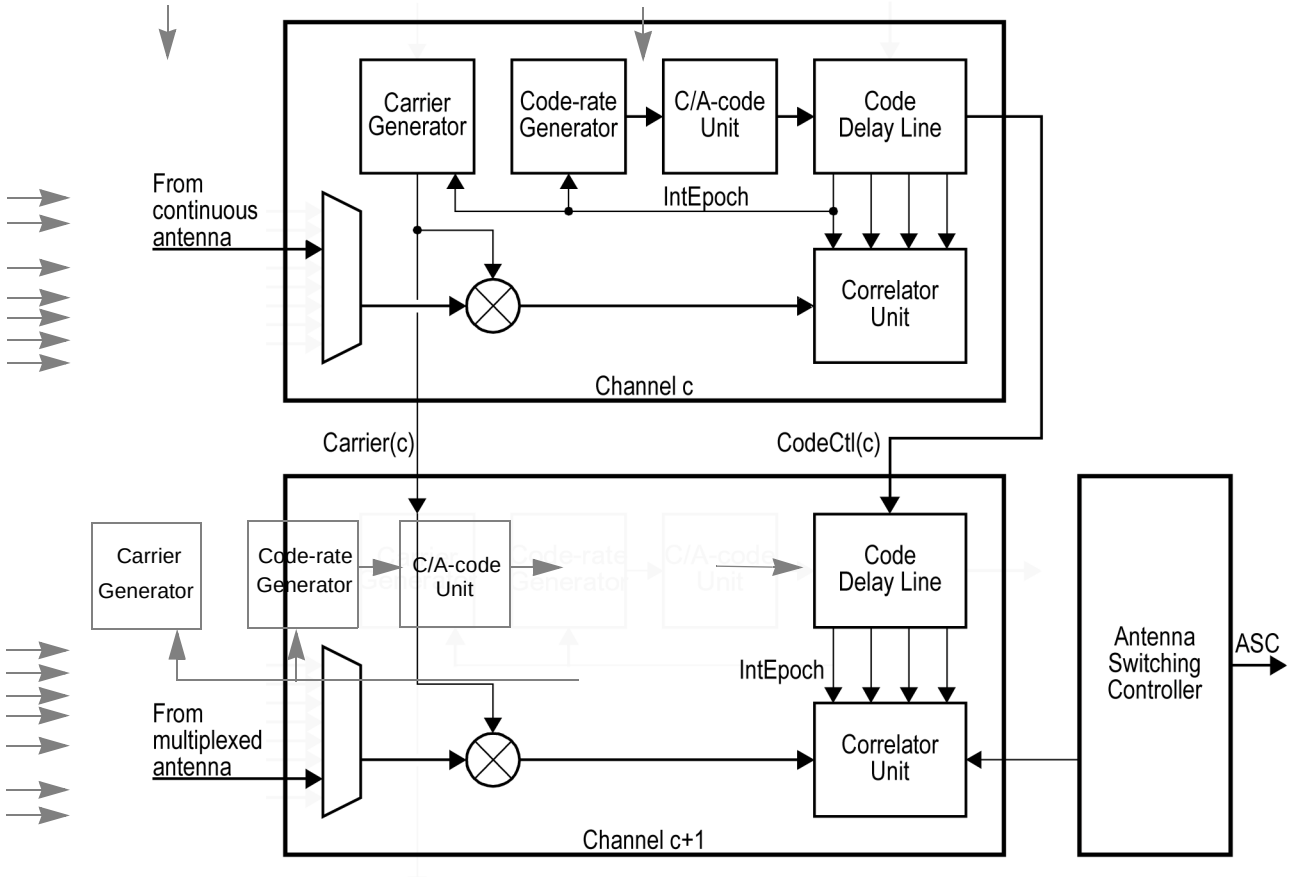
The AGGA-2 can slave single-frequency channels for attitude determination, fast acquisition and multipath identification. Within one AGGA-2, channels processing C/A-code can be slaved to each other as shown in Figure 5. The signals used for slaving are the Carrier *(Carrie*r) and the Code & Control *(CodeCt*l). For any channel *c (c* in 0 to 11) these signals can be selected from the preceding channel *c*-1, with wraparound at 0 and at 11. Slaving for single-frequency operation is described in page 17 and page 18.

One CA-channel together with two CaP-channels and one P-code Unit can be slaved into a dual-frequency channel, which is shown as one column of the Channel Matrix in Figure 5. Slaving for dual-frequency operation is described in page 19.

The slaving of each channel can be programmed by the *CodeSel* and *CarrierSel* fields of the *ChannelMode* register, including the selection of P-code & Control signals for a

CaP channel. When a unit is bypassed due to channel slaving, it is partly powered down to reduce power consumption. Furthermore, each channel can be individually enabled by the *ChannelOn* bit of the *ChannelMode* register. When a channel is not enabled it is partly powered down. See page 67 for power reduction programming aspects.

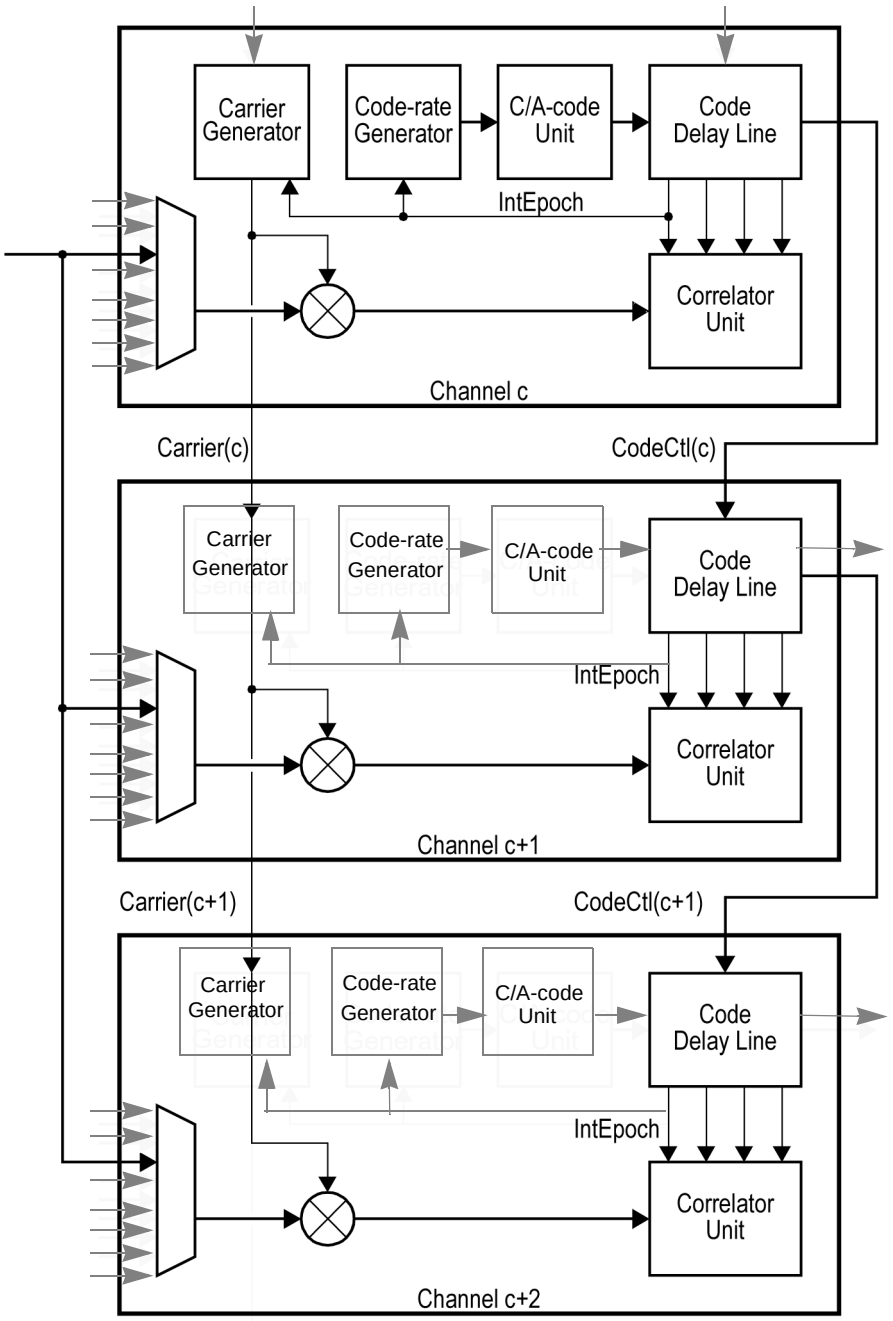
Figure 8. Two channels slaved for Hybrid Parallel Multiplex attitude determination



Channel Slaving for Attitude Determination

Figure 8 shows an example with two channels processing C/A-code slaved for attitude determination using the Hybrid Parallel-Multiplex scheme. The Input Selectors are selecting signals from different antennas. The *Carrier* for both channels is generated by the Carrier Generator of channel c. Similarly, the Code & Control signals for both channels are generated by the Code-rate Generator and C/A-code Unit of channel c. Bypassed units are greyed out in the figure. The Code Delay Line of channel *c* is programmed for zero delay between the two channels so that the C/A-code sequence has the same phase in both channels. The Antenna Switching Controller controls the integration in the Correlator Unit by the Antenna Switch Epoch strobe, synchronously with the switching of an external antenna multiplexer (not shown in the figure) controlled by the *ASC* signal, as further described page 32**.**

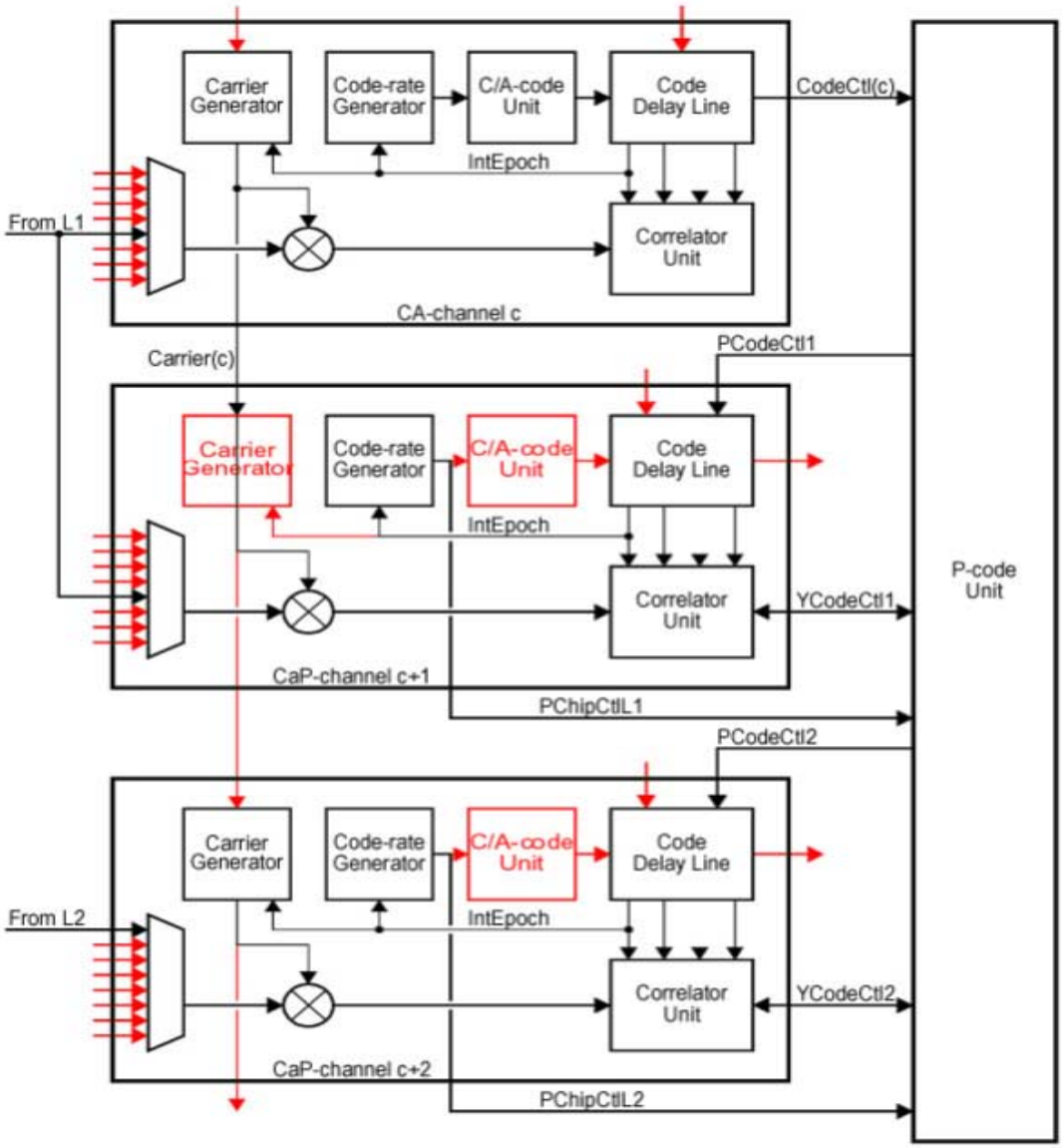
Figure 9. Three channels slaved for fast acquisition or mulitpath identification



Channel Slaving for Fast Acquisition and Multipath Identification

Figure 9 shows an example with three channels processing C/A-code slaved for fast acquisition or multipath identification. The Input Selectors are set to the same Front-end input. The *Carrier* for all channels is generated by the Carrier Generator of channel c. Similarly, the Code & Control signals for all channels are generated by the Code-rate Generator and C/A-code Unit of channel c. Bypassed units are greyed out in the figure. The Code Delay Lines of each channel provides despreading signals with an adequately spaced C/A-code sequence. For fast acquisition the code spacing is typically 1/2 or 1 chip while for multipath identification it might be in the order of 0.1 chip. Details on the selection of code spacing can be found in page 28. In this example the incoming signal is correlated by nine complex despreaders and integrators.

Figure 10. Three channels slaved for dual frequency operation



Channel Slaving for Dual-frequency Operation

Figure 10 shows one CA-channel together with two CaP-channels and one P-code Unit slaved as a dual-frequency channel capable of tracking C/A-code on one channel and P-code or Y-code on two channels. Each P-code Unit generates one replica P-code sequence, a delayed version of this P-code sequence, and the control strobes required for dual-frequency operation including semi-codeless operation.

As shown in the figure the CA-channel *c* provides the *Carrier* for the L1 CaP-channel. However, the L1 CaP-channel may instead use its internally generated *Carrie*r. The Integration Epoch for the entire dual-frequency channel is generated by the CA-channel and provided via the *PCodeCtl* signal so that all Correlator Units of a dual-frequency channel integrate over the same period. This Integration Epoch is also used for the P-code handover after initialisation (see page 34). The P-code chip rates for the L1 and L2 signals are generated by the Code-rate Generators of the respective CaP-channels.

The two *YCodeCtl* signals provided by the P-code Unit allows the CaP-integrators in the Correlator Units of the CaP-channels to operate with GPS Y-code (encrypted P-code) using the Adaptive Semi-codeless technique, as described in the next section.

Adaptive Semi-codeless Dual-frequency Operation

The GPS Y-code is generated by multiplying the known P-code sequence with an unknown W-code sequence. The length of one W-code chip is assumed to be an integer number of P-code chips, where each W-code chip can have the value -1 or +1. The resulting Y-code sequence can thus be seen as a suite of short sequences of the original P-code, with some of these short sequences being inverted. The purpose of the Adaptive Semi-codeless processing implemented in the AGGA-2 is to allow correlation of the Y-code signal using estimates of the W-code chips, without knowing the actual W-code sequence.

When operating in semi-codeless mode, the CaP-integrators for the L1 signal and for the L2 signal first integrate the baseband signal despread with the selected P-code for a period assumed to correspond to one W-code chip. Since the thermal noise on the L1 and L2 signals is statistically independent, the primary correlation value for the L1 signal can used to estimate the value of the W-code chip for the L2 signal, and vice versa. For both the L1 and L2 signals the primary correlation values are subsequently accumulated taking into account the value of the W-code chip as estimated from the primary correlation value of the other channel. By virtue of an adjustable threshold, primary correlation values for which the corresponding W-code estimations are indecisive can be excluded from this secondary accumulation. This two-stage integration is performed until the end of the Integration Epoch when the results of the secondary accumulations are latched into the correlation registers.

Enabling the W-edge Generator in the Semi-codeless Controller in the P-code Unit configures the corresponding dual-frequency channel for semi-codeless operation, as detailed in page 38.

Note: A patent application has been filed for the Adaptive Semi-codeless processing technique. Users intending to use this technique are advised to contact the Patent Office at ESA Headquarters for further information.

Input Selector

The Input Selector selects one of the eight *IQ* signals from the Front-end Interface. Each *IQ* signal provides a complex sample every *CoreClk* cycle, consisting of an I component and a Q component. Each component is represented using the internal format corresponding to the values [-3, -1, +1, +3]. Which *IQ* signal is used is selected by programming the *InputSel* field of the *ChannelMode* register.

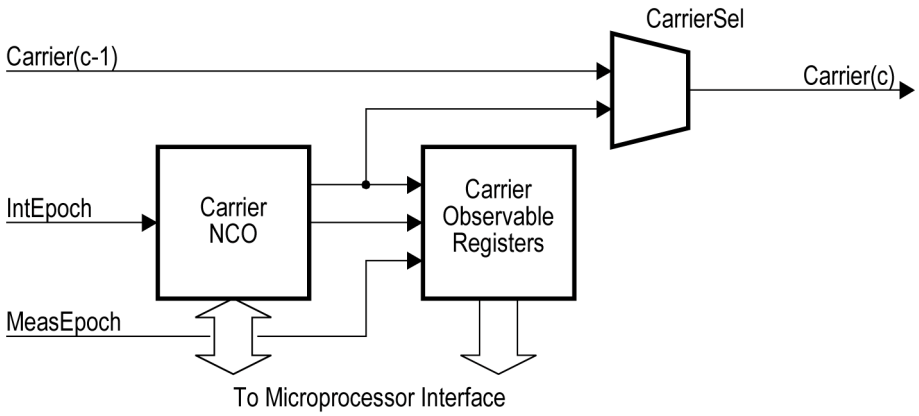
Carrier Generator

The Carrier Generator shown in Figure 11 generates the replica Carrier. It consists of the phase-accumulator type Carrier NCO and the Carrier Observable Registers. The *Carrier* is generated by adding a signed phase increment to the Carrier NCO phase register every *CoreClk* cycle. The frequency range is ±*CoreClk*/2 with a resolution of *CoreClk*/12·2 ^{28}, corresponding to ±1.5 MHz and 0.0009 Hz for a *CoreClk* frequency of 3 MHz and to ±15 MHz and 0.0093 Hz for a *CoreClk* frequency of 30 MHz. Both the frequency and phase can be programmed, with a new value becoming effective at the start of the next Integration Epoch. The four MSBs of the Carrier NCO phase register are output as the *Carrier* signal, with twelve possible values ranging from -6 to +5. The value represents a fraction of the carrier cycle, expressed as the number of twelfths of a full carrier cycle. The *Carrier* is used by the Final down-converter.

The Carrier Cycle Count and part of the Carrier Phase are sampled at the end of each Measurement Epoch so they can be read by the microprocessor. This observable can be used for velocity calculation, for attitude determination and to smooth the pseudorange.

Whether the internally generated *Carrier* or the *Carrier* from another channel is used for the final down-conversion is selected by the *CarrierSel* field of the *ChannelMode* register. When an external *Carrier* is used the value of the Carrier Observable becomes undefined. The Carrier Generator could then also be powered down, see page 67.

Figure 11. Carrier Generator block diagram



Carrier NCO

The 32-bit Carrier NCO generates the *Carrier* signal by adding a signed phase increment to the Carrier NCO phase register every *CoreClk* cycle. This phase increment is programmed by the *CarrierFreq* register, as a two's complement signed value representing phase increments between -180° and 180°. The *Carrier* frequency f_{ca} is obtained by programming the *CarrierFreq* register with the value $12 \cdot 2^{28} \cdot f_{ca} / f_c$, where f_c is the *CoreClk* frequency. A newly programmed value becomes effective at the start of the next Integration Epoch.

The *Carrier* phase can be directly adjusted by adding a phase shift to the Carrier NCO phase register. This phase shift is programmed by the *CarrierShift* register, and a newly programmed value is added once at the beginning of the next Integration Epoch. The phase shift is represented as a 12-bit two's complement signed value which is added to the 12 MSBs of the Carrier NCO phase register. The valid range of *CarrierShift* is from -1536 to 1536, corresponding to phase shifts of up to ±180° with a resolution of 0.12°. This resolution is the same as for the *CaPhase* field of the Carrier Observable.

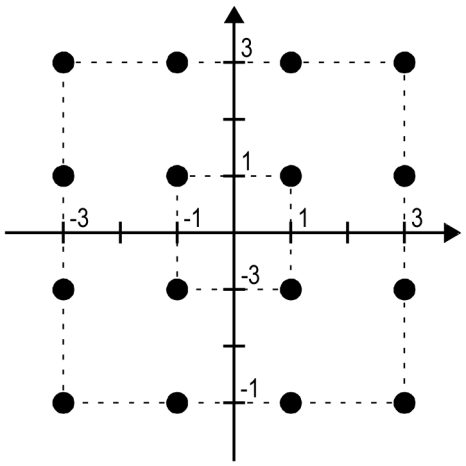
The 32-bit Carrier NCO counts modulo $12 \cdot 2^{28}$, where a counter wraparound indicates the end of a complete carrier cycle. A wraparound in the positive direction increments

the 12- bit Carrier Cycle Count in the Carrier Observable Registers whereas a wrap-around in the negative direction decrements it. The Carrier Cycle Counter operates in a wrap-around fashion, only preserving the LSBs of the count. The four MSBs of the Carrier NCO phase register are output as the *Carrie*r, having twelve possible values from -6 to +5.

Carrier Observable Registers

At the end of each Measurement Epoch the 12-bit Carrier Cycle Count *CaCycleCount* and the 12 MSBs of the Carrier Phase *CaPhase* are sampled and placed in the Carrier Observable *CarrierOb*s. The resolution of *CaPhase* is 0.12°, corresponding to 0.06 mm at the L1 frequency. The Carrier Cycle Count range is from -2048 to 2047.

Figure 12. Complex signal constellation



Note: Since the Carrier Cycle Count only represents the LSBs of the actual number of carrier cycles, in some cases, such as for establishing Doppler compensation, the firmware must calculate the total number of carrier cycles.

Final Down-converter

The Final Down-converter removes the residual carrier from the complex input signal selected from the Input Selector by subtracting the angle represented by the *Carrier* signal from the Carrier Generator. The residual carrier can include the IF after Real-to-Complex conversion, GLONASS channel frequency, Doppler shift and local oscillator frequency offset. After the Final Down-converter the complex signal vector should have a residual carrier frequency sufficiently close to 0 Hz to have negligible impact on the correlation process. The down-conversion frequency range is determined by the Carrier Generator, and has a range of $\pm CoreClk/2$ (see page 21). The complex input and output signals use the internal representation corresponding to the values [-3, -1, +1, +3] for each of the I and the Q components. The down-converted baseband signal is provided to the Correlator Unit.

As shown in Figure 12, the sixteen possible combinations of the I and Q values can be represented as four points on an inner ring and twelve points on an outer ring. Down-conversion is performed by rotating the complex input signal by an angle corresponding to the *Carrier* signal, in a clockwise direction for positive *Carrier* values. The *Carrier* value represents the number of 30° fractions of a full carrier cycle, ranging from -6·30° to 5·30°. Since there are only four points on the inner ring, the rotation angle is rounded to the nearest 90° for samples on the inner ring. Since the points on the outer ring are not placed with exactly 30° spacing, the rotation angle is rounded to be aligned with the nearest point on the outer ring.

Code-rate Generator

The Code-rate Generator shown in Figure 13 generates the code chip rate and the associated control signals. It consists of the phase-accumulator type Code NCO and the Code Phase Observable Register. The chip rate is generated by adding an unsigned phase increment to the Code NCO phase register every *CoreClk* cycle. The frequency range is from 0 to *CoreClk*/2 with a resolution of *CoreClk*/2 32, corresponding to 0 to 1.5 MHz and 0.0007 Hz for a *CoreClk* frequency of 3 MHz, and 0 to 15 MHz and 0.007 Hz for a *CoreClk* frequency of 30 MHz. Both the chip rate and the code phase can be programmed, with a new value becoming effective at the start of the next Integration Epoch. Alternatively, the code shift can be programmed to be applied immediately, or repeatedly for every Integration Epoch. Two shift step sizes are supported; one for normal acquisition and one, coarser and thus faster, shift step size intended for fast acquisition with many slaved channels and short integration times.

The Code NCO generates two clocks grouped into the *ChipCtl* signal; one at the chip rate and one at double the chip rate. The chip rate clocks the C/A-code Unit, or for the case of a CaP-channel in dual-frequency configuration it clocks the P-code Unit. The signal at double the chip rate can be used for code spacing in the Code Delay Line.

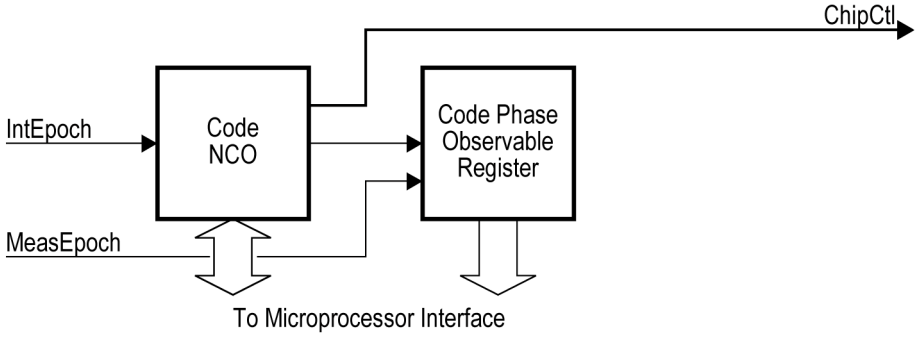
Part of the Code NCO phase is sampled at the end of each Measurement Epoch so it can be read by the microprocessor. This observable can be used for calculating a pseudorange.

In slaving configurations where *ChipCtl* is not used (see page 16) the value of the Code Phase Observable becomes undefined. The Code-rate Generator could then also be powered down, see page 67.

Code NCO

The Code NCO generates the chip rate by adding an unsigned phase increment to the Code NCO phase register every *CoreClk* cycle. The code frequency (or chip rate) f_{co} is obtained by programming the *CodeFreq* register with the unsigned value $2^{32} \cdot f_{co} / f_c$, where f_c is the *CoreClk* frequency. A newly programmed value becomes effective at the start of the next Integration Epoch.

Figure 13. Code-rate Generator block diagram



Note: When the chip rate is higher than one-quarter of the *CoreClk* rate, the signal at double the chip rate becomes invalid and should not be used for code spacing in the Code Delay Line.

Note: The *CodeFreq* register should not be set to a value lower than a chip rate of 1/64th of the *CoreClk* frequency when negative code shifts are used, since this can cause undefined behaviour. For the same reason, it should not be set to a value higher than (1/ 2-1/64) ˙*CoreClk* or (1/2-1/8) ●*CoreClk* if positive code shifts are used (which value

Note: applies depends on the magnitude of the selected code shift step).

Warning: Programming the *CodeFreq* register to zero means that it will not be possible to reprogram the register, since an Integration Epoch will not occur and thus a new value will never become effective. The code phase can be adjusted by a number of code shift steps which are added to the Code NCO phase register by one code shift step per *CoreClk* cycle.

The code phase shift is programmed by the *CodeShiftCtl* register, as a 10-bit number of code shift steps and information on whether the shift is in positive or negative direction (in principle a sign/ magnitude representation). The range of code shift steps that can be applied is from -1023 to 1023, as determined by the unsigned field *ShiftSteps* and the *ShiftDir* bit. The *StepSize* bit selects the code shift step to be either 1/64th or 1/8th of a code chip, the latter step size only being valid for positive code shifts. With a step size of 1/64th of a code chip the code shift range is ±16 chips with a resolution of 1/64th of a chip, whereas with a step size of 1/8th of a code chip the code shift range is 0 to 128 chips with a resolution of 1/8th of a code chip. The resolution is lower than what is available in the *CodePhase* field of the Code Observable; higher precision has to be achieved by fine-tuning the code frequency.

The code phase shift can be applied immediately when writing a new value to the *CodeShiftCtl* register, or alternatively it can be deferred until the start of the next Integration Epoch. In addition, the Code NCO can be programmed to automatically apply the phase shift at the start of every Integration Epoch, allowing automatic code shifting during code acquisition. The code shift mode is programmed by the *ShiftMode* field.

Note: Only one phase shift command can be in progress at any given time; no other phase shifts are accepted while one phase shift operation is in progress, which can take up to 1024 *CoreClk* cycles.

The 32-bit Code NCO counts modulo 2^{32}, where a counter wraparound indicates the end of a complete code cycle, which is signalled by the chip rate element of *ChipCt*l.

Code Phase Observable Register

At the end of each Measurement Epoch the 16 MSBs of the code phase *CodePhase* are sampled and placed in the *CodePhase* field of the Code Observable register *CodeObs* (see also page 26). The resolution of *CodePhase* is 0.000015 code chips which corresponds to 5 mm for the GPS C/A-code and 9 mm for the GLONASS C/A-code.

C/A-code Unit

The C/A-code Unit shown in Figure 14 generates the C/A-code and associated signals. It consists of the C/A-code Generator, the Integration Timer and the Code Chip Observable Registers. The C/A-code Generator can generate any GNSS C/A-code. The code rate is determined by the Code-rate Generator, which provides one clock at the chip rate and one at double the chip rate grouped as the *ChipCtl* signal. The Integration Timer generates the Integration Epoch strobe which can have a period from 1/4th of a C/A-code epoch up to 20 C/A-code epochs. The C/A-code sequence and the two associated control signals for the double chip rate and the Integration Epoch are grouped into the *CACodeCtl* signal which is provided to the Code Delay Line.

The number of C/A-code epochs since the last Integration Epoch and the number of C/A-code chips since the start of the last C/A-code epoch are sampled at the end of each Measurement Epoch so they can be read by the microprocessor. This observable can be used for calculating a pseudorange and for solving the C/A-code cycle ambiguity.

In slaving configurations where the C/A-code Unit is not used (see page 16) it is automatically powered down, and the related observables become undefined.

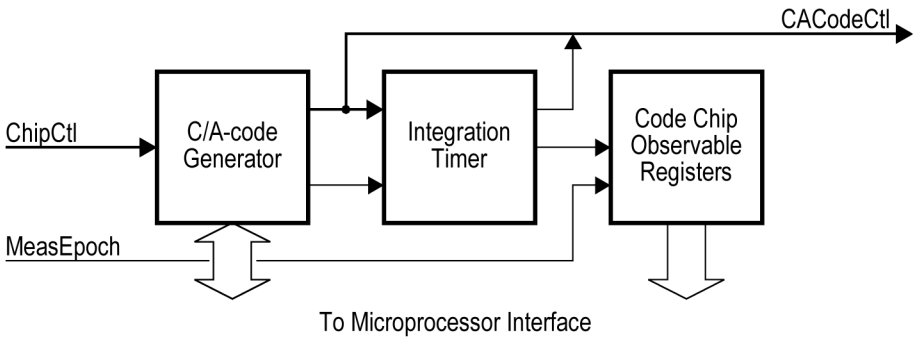
Note: When the internal C/A-code Unit is used (i.e. the code is not slaved), the Integration Epoch generated by this unit is the one that is used by the entire channel, provided via the Code Delay Line.

C/A-code Generator

The C/A-code generator generates the GNSS replica C/A-code sequence, with the chip rate being determined by the Code-rate Generator. In addition to the replica C/A-code sequence, the C/A-code generator signals a C/A-code epoch each time the C/A-code period has elapsed, as defined by the all-one state of the GLONASS or GPS G1 code generator shift register. The C/A-code sequence forms part of the *CACodeCtl* signal.

Note: Programming the *CodeSetting* register resets the C/A-code Generator to its initial state, but does not affect the state of the Integration Timer. The first observables and correlation values after programming this register will therefore be invalid.

Figure 14. C/A-code Unit block diagram



C/A-code Generator in GLONASS mode

The GLONASS C/A-code sequence has a length of 511 chips. It has a chip rate of 511 kHz and a code epoch of exactly 1 ms, as defined in RD2. The code sequence is taken from the seventh tap of a 9-stage LFSR with the generating polynomial:

$$G = 1 + X^5 + X^9$$

Generation of the replica GLONASS C/A-code is programmed by the *GPSMode* bit of the *CodeSetting* register. At the same time the C/A-code Generator is reset to its initial state (all ones).

C/A-code Generator in GPS mode

The GPS C/A-code sequence has a length of 1023 chips. It has a chip rate of 1023 kHz and a code epoch of exactly 1 ms, as defined in RD1. The code sequence is generated by the modulo-2 addition of the outputs of two 10-stage LFSRs with the generating polynomials:

Different GPS-type C/A-code sequences can be generated by programming different initial states for the G2 sequence. The generation of the replica GPS-type C/A-code sequence is programmed by the *GPSMode* bit and the *InitG2* field of the *CodeSetting* register. At the same time the C/A-code Generator is reset to its initial state (G1 being all ones, G2 having the programmed value). Any GNSS C/A-code sequence except GLONASS C/A-code can be generated when the C/A-code Generator is in GPS mode.

Integration Timer

When the C/A-code generator is used the C/A-code Integration Timer generates the Integration Epoch strobe with a period of 1/4, 1/2, 1, 2, 5, 10 or 20 C/A-code epochs, as programmed by the *EpochLen* field of the *IntEpochLen* register. Each time the Integration Timer has reached zero it is reset and restarted, and an Integration Epoch interrupt is generated to inform the microprocessor that new correlation values are available from

the Correlator Unit. The Integration Epoch strobe is output as part of the *CACodeCtl* signal to the Code Delay Line, from which it is further distributed within the channel and to any slaved channel. A newly programmed value becomes effective at the next Integration Epoch coinciding with a full C/A-code Epoch (to ensure synchronisation with the C/A-code Epochs).

Optionally, a dead-time window can be specified before the end of each Integration Epoch, with the width being an integer number of C/A-code Epochs, as programmed by the *EpochDeadTime* field of the *IntEpochLen* register. At the start of the dead-time window the correlation values are latched into the correlation registers and the corresponding interrupt is generated. However, newly programmed values for the Carrier and Code-rate Generators do not become effective until the end of the Integration Epoch (which is also the end of the dead-time window). This gives the firmware the possibility to calculate and provide new carrier and code parameters already for the next Integration Epoch. Using the dead-time features allows better control loop performance due to the shorter control delay, but it requires fast microprocessor interrupt response time in the order of some milliseconds. Using the dead-time feature also shortens the effective integration interval leading to some reduction of the processing gain of the GNSS signals.

Code Chip Observable Registers

At the end of each Measurement Epoch a counter value representing the number of C/A-code epochs since the start of the current Integration Epoch *CodeEpochCnt* and the number of C/A-code chips since the start of the current C/A-code epoch *ChipCount* are sampled and placed in the Code Observable register *CodeObs* (see also page 24).

Note: The total number of C/A-code epochs should be counted by the firmware to form the complete pseudorange.

Note: When the Integration Timer is programmed for 1/4, 1/2 or 1 C/A-code epoch, as can be useful during acquisition, the Code Epoch Count is invalid.

Code Delay Line

The Code Delay Line shown in Figure 15 selects which Code & Control signals shall be used in the channel, produces the Early *(*E), Punctual *(*P) and Late *(*L) versions of the selected code, and distributes the Integration Epoch strobe *(IntEpoc*h) corresponding to the selected code sequence within the channel. Whether the internally generated *CACodeCtl* or the slaving *CodeCtl* signal is used in the channel is selected by the *CodeSel* field of the *ChannelMode* register. In the case of a CaP-channel it is also possible to select the *PCodeCtl* signal from the P-code Unit. The selected Code & Control signal is output as the *CodeCtl(c)* signal to be used for subsequent slaving.

The E, *P* and *L* phased code sequences are selected from a 13-stage shift register which is fed with the selected code sequence. The phase delay between two adjacent shift register taps is referred to as one space. The output code sequences are spaced by multiples of half a code chip or multiples of *CoreClk* cycles. The spacing between the *E* and the P, and between the *P* and the L, code sequences is independently programmable. The three phased code sequences are provided to the Correlator Unit for despreading the base-band signal from the Final down-converter.

When channels are slaved, the Code Delay Lines of several channels are cascaded. For fast acquisition and multipath identification the Code Delay Lines are configured to produce many delayed versions of the code sequence so that many correlators use phase delayed versions of the same code sequence. For attitude determination the correlators of the slaved channels should use the same code sequence without any delay. The Spacing Sequencer supports time-multiplexed multipath identification techniques.

Code Spacing Selection

Normally the spacing between the *E* and the *P* code sequences is determined by the *ChipSpacingEP* field, and the spacing between the *P* and the *L* code sequences is determined by the *ChipSpacingPL* field of the *CorrMode* register. The spacing can be selected between one, two, four or six *CoreClk* cycles, or alternatively half a chip, one chip, two chips or three chips, depending of the value of the *SpaceSel* bit.

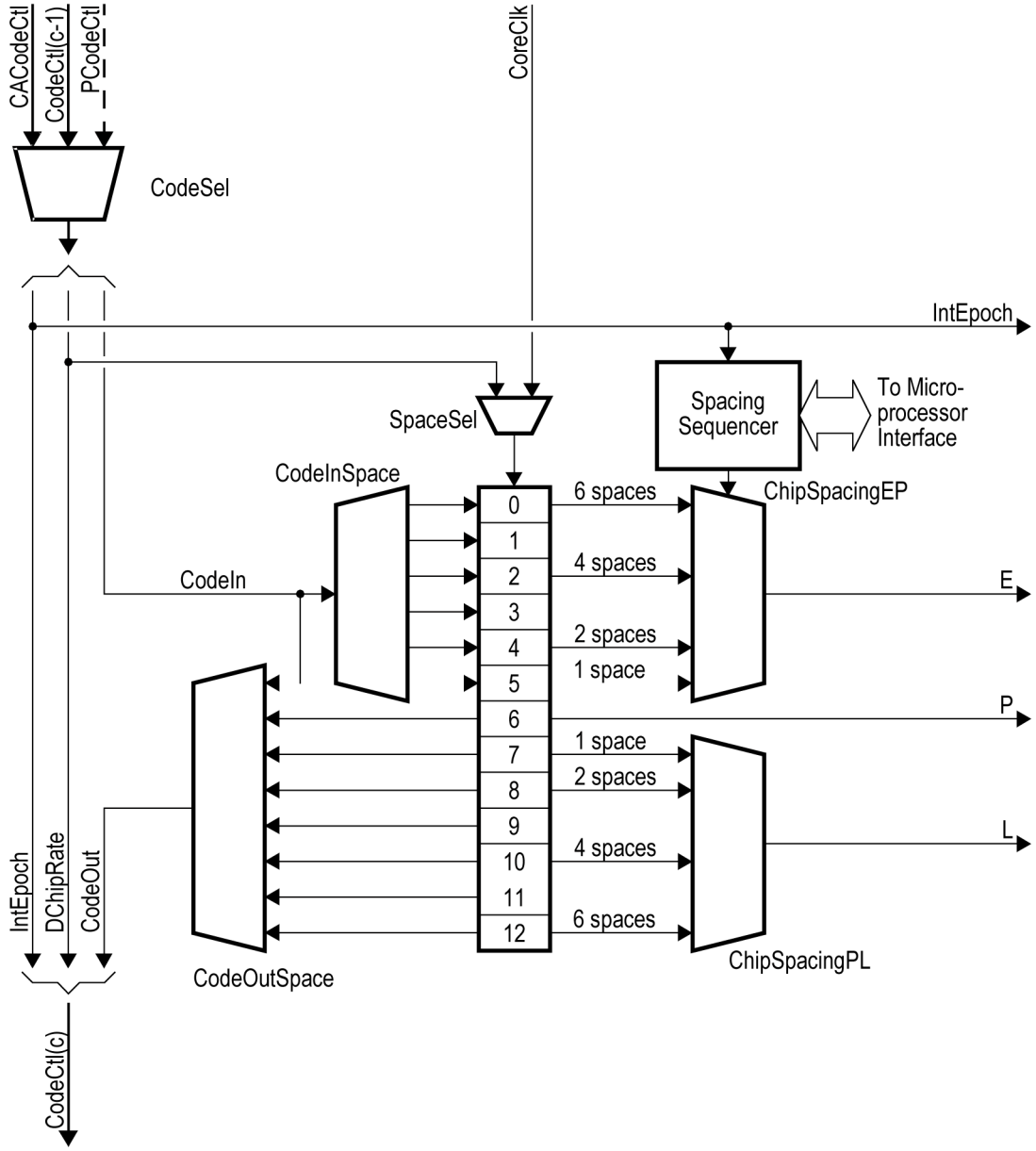
Note: The double chip rate signal is only valid for code chip rates lower than one quarter of the *CoreClk* rate (see page 23)**.** For higher chip rates (always for GPS P-code), the code spacing must be programmed in numbers of *CoreClk* cycles.

The spacing from the incoming code *CodeIn* to *P* can be programmed by the *CodeInSpace* field of the *CorrMode* register and ranges from two spaces to seven spaces. For proper operation there should never be less than one space between *CodeIn* and E. *CodeOut* is a delayed version of the code sequence to be used for channel slaving. The spacing between *P* and *CodeOut* is programmed by the *CodeOutSpace* field of the *CorrMode* register and ranges from zero to six spaces. In addition, it is possible to program zero delay between *CodeIn* and *CodeOu*t, to be used for attitude determination (see page 17) and when the GNSS code is output (see page 47).

The Spacing Sequencer can be programmed to automatically change the code spacing between *E* and *P* for each new integration period, allowing to measure the correlation profile without the need of channel slaving. This information can then be used for multipath identification. When enabled, a new spacing between *E* and *P* is selected at the beginning of every Integration Epoch. Two to four spacing settings in sequence can be programmed by the *SwitchPeriod* field of the *CorrMode* register. The spacing for each period is then programmed by the corresponding *SwitchId0* (same as *ChipSpacingE*P) to *SwitchId3* fields. Which spacing used during a particular integration period can be read from the *SwitchId* field.

Note: Sequencing the code spacing might not be suitable for applications with high dynamics.

Figure 15. Code Delay Line block diagram



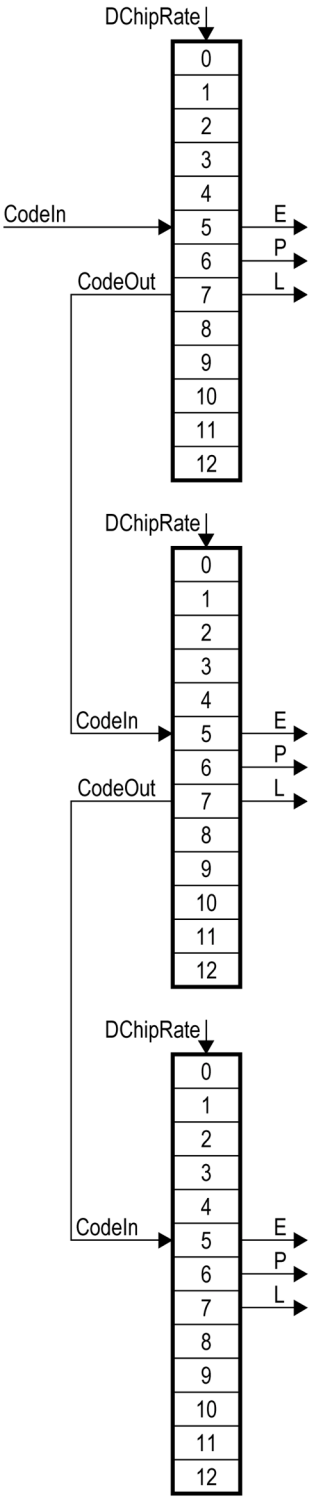
Examples of Code Spacing for Channel Slaving

By connecting the output of one Code Delay Line to the input of the Code Delay Line of a slaved channel a single long delay line with many phased code sequence outputs can be formed. In Figure 16 two examples are shown, for fast acquisition and for multipath identification. In Figure 15 a) a configuration for C/A-code fast acquisition with three channels is shown, with the spacing programmed in multiples of half chips. Nine complex integrators are configured to correlate the nine phased code sequences spaced by 0.5 code chips. In Figure 15 b) a configuration for multipath identification with three channels is shown. Nine correlators are configured to correlate nine code sequences

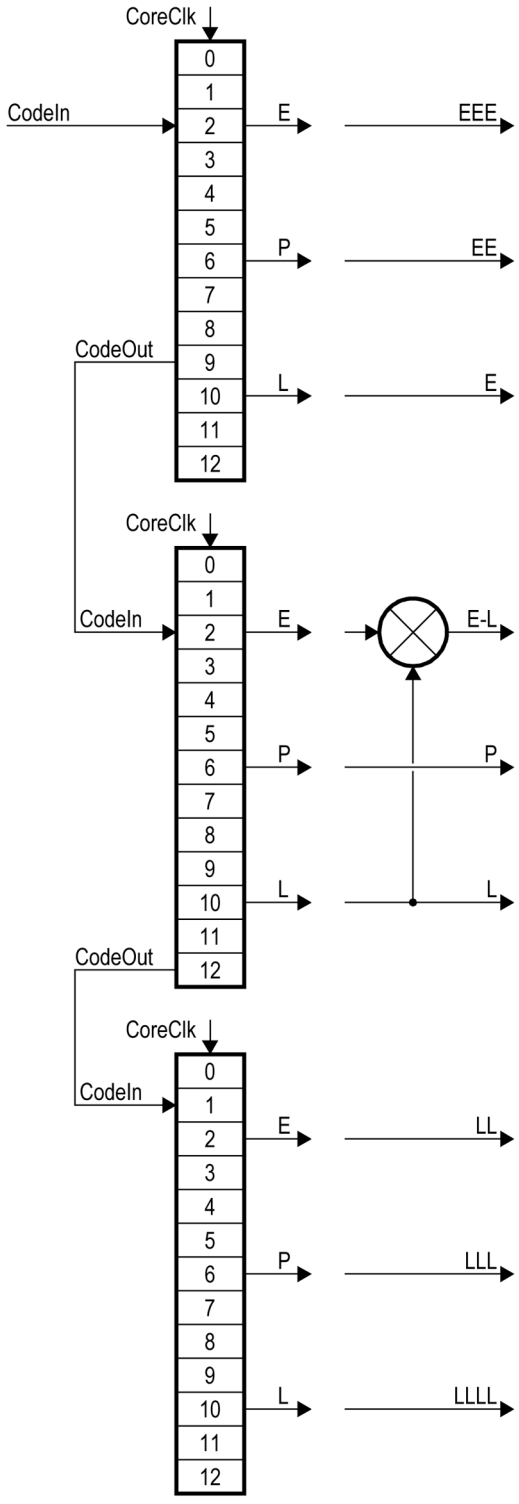
being EEE, EE, E, E-L, P, L, LL, LLL and LLLL, with a spacing of four *CoreClk* cycles. The *E-L* operation shown to the right in the figure is performed in the Correlator Unit.

Note: The reference to “multipath identification” in the slaving refers to that the AGGA-2 hardware supports various code spacing possibilities.
However, to achieve multipath identification also requires appropriate multipath models and identification algorithms, which may not always be possible to establish.

Figure 16. Two examples of three channels with slaved Code Delay Lines



a) Slaving for fast acquisition



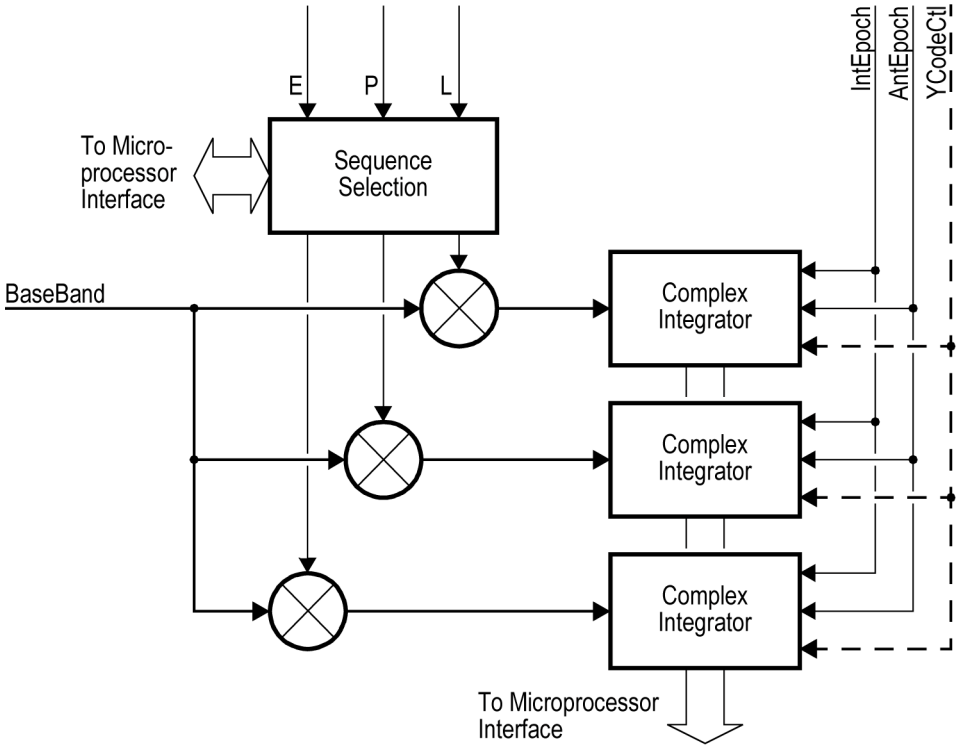
b) Slaving for multipath identification

Correlator Unit

The Correlator Unit shown in Figure 17 consists of three signal despreaders, a Sequence Selection module and three complex integrators. The Correlator Unit of a CA-channel contains CA-Integrators while for a CaP-channel it contains CaP-Integrators. The complex *BaseBand* signal from the Final Down-converter is multiplied with three code sequences. Each integrator integrates a despread baseband sequence during one Integration Epoch, after which the correlation values are latched into the correlation registers. These correlation values can be used by the receiver firmware to close the tracking loops, including carrier and code phase error computation, application of loop filters, estimation of signal and noise amplitude, etc.

The Antenna Switch Epoch *AntEpoch* strobe controls the integrator operation for Hybrid Parallel-Multiplex attitude determination. The CaP-integrators can, in addition to processing C/A-code, also process P-code and GPS Y-code (encrypted P-code), the latter supported by the *YCodeCtl* signals from the P-code Unit.

Figure 17. Correlator Unit block diagram



Sequence Selection

Seven different sets of code sequence combinations can be selected for despreading the *BaseBand* signal by programming the *DespreadMode* field of the *CorrMode* register. Each set is a combination of code sequences of the three spreading code sequences E, *P* and *L* from the Code Delay Line. The possible combinations and their intended usage are shown in Table 2.

Table 2. Despreading sequence modes

Integrator 0	Integrator 1	Integrator 2	Description
P	E-L	Off	For tracking when separate E or L correlation values are not needed
P	E	L	For fast acquisition, for measuring the absolute E, *P* and *L* correlation values for multipath identification, and for tracking with separate *E* and *L* correlation values
P	E-L	E	For tracking with multipath identification, providing a separate *E* correlation value
P	E-L	L	For tracking with multipath identification, providing a separate *L* correlation value
P	P	Off	For Hybrid Parallel-Multiplex attitude determination for the slaved channel, only using the *P* sequence with special updating
P	Off	Off	For Parallel attitude determination for the slaved channel, only using the *P* sequence
B	Off	Off	For calibration, with no despreading and only one integrator being used
Off	Off	Off	To save power when channel is not used

"Off" indicates that the corresponding integrator is not used and powered down, and the value of the corresponding correlation values are undefined. "B" indicates that the despreader is bypassed. "E-L" is defined as the L sequence subtracted from the E sequence, with the result of this subtraction being divided by two. The signals after despreading can have the values [-3, -1, 0, +1, +3] for each of their I and the Q components, with the value zero present only for the case of "E-L" despreading.

Note: For "E-L" despreading the resulting correlation values are divided by two, as a consequence of the division by two as described above.

CA-Integrator

Each of the two 22-bit accumulators for the I and the Q components in a CA-Integrator accumulates the despread base-band signal over one Integration Epoch. Each *CoreClk* cycle the value after despreading [-3, -1, 0, 1, 3] is added to the correlation sum. At the end of each Integration Epoch the correlation values for that channel are latched into the six correlation registers *CorrValue0*I, *CorrValue0*Q, *CorrValue1*I, *CorrValue1*Q, *CorrValue2I* and *CorrValue2*Q, and the sum is reset to zero.

To identify a situation where the previous values were not read out in time, the *OverRun* bit in the *CorrValue0I* register is set when the *CodeFreq* register for the same channel was not written before a new Integration Epoch occurred. The *OverRun* bit thus indicates that new control parameters where not available at the start of the next Integration Epoch, when they should have become effective.

The Correlator Unit can be programmed for Hybrid Parallel-Multiplex attitude determination operation by selecting the (P, P, Off) despreading setting. In this mode Integrator 0 updates its correlation registers at the end of each Integration Epoch, while integrator 1 updates its correlation registers at the end of each Antenna Switch Epoch. Both integrators are reset to zero both at the start of each Integration Epoch and at the start of each Antenna Switch Epoch. To reduce the impact of noise due to antenna switching, the accumulation is inhibited as long as the *ASEI* input is asserted. The firmware might need to compensate for the time the accumulation is inhibited.

Note: When *ASEI* is driven by *ASE*O, the accumulation is inhibited for 128 *CoreClk* cycles (for example, corresponding to nine GPS code chips at a *CoreClk* frequency of 15 MHz). Different inhibition lengths can be obtained by changing the time the *ASEI* input is asserted using external circuitry.

CaP-Integrator

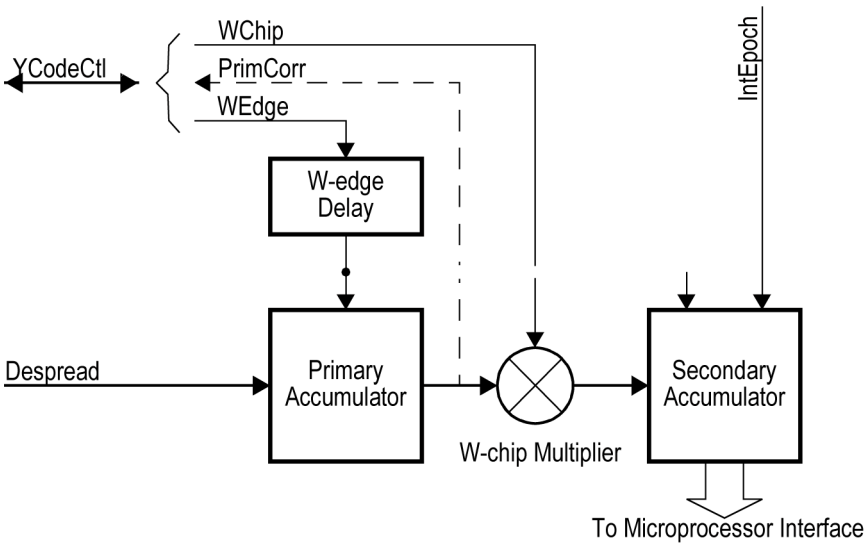
A CaP-integrator can integrate C/A-code, P-code and GPS Y-code (encrypted P-code) modulated signals. When processing C/A-code the CaP-integrator behaves identically to the CA-integrator described in page 25. For P-code processing the only difference is that the P-code sequence has been selected in the Code Delay Line. For Y-code operation an adaptive semi-codeless processing technique is employed, controlled by the Y-code Control *(YCodeCtl*) signal provided by the Semi-codeless Controller in the P-code Unit, as described in the following section.

CaP-Integrator in Adaptive Semi-codeless Mode

The CaP-integrator configured for semi-codeless operation shown in Figure 18 allows processing of the GPS Y-code, supported by the Y-code and Control *(YCodeCtl*) signal provided by the Semi-codeless Controller in the P-code Unit. It consists of a 9-bit Primary Accumulator, the W-chip Multiplier and a 22-bit Secondary Accumulator, all operating with complex signals. *YCodeCtl* consists of the *WEdge* signal indicating the assumed limits of the W-code chips and the *WChip* signal providing the estimated value of the W-code chip, with the possible values [-1, 0, 1]. Either the I or the Q component of the primary correlation result from the Punctual CaP-integrator (integrator 0) is provided to the Semi-codeless Controller, to be used for the estimation of the W-code chip for the other channel. For the L1 CaP-channel *PrimCorr1* is taken from the Q component, since the firmware normally tracks the L1 C/A-code in the I correlator the P-code correlation values thus appear in the Q component. For the L2 CaP-channel *PrimCorr2* is taken from the I component, and the firmware should therefore track the L2 P-code in the I correlator.

The despread baseband signal multiplied with the estimated W-code sequence is integrated by the Secondary Accumulator during one Integration Epoch, after which the correlation values are latched into the correlation registers. These correlation values can be used by the receiver firmware for carrier and code phase error computations, etc.

Figure 18. CaP-Integrator configured for semi-codeless operation



Primary Accumulator

The 9-bit Primary Accumulator integrates the baseband signal despread by the replica P-code sequence over one W-code chip as signalled by *WEdg*e. At the end of each W-code chip the Primary Accumulator results are provided to the W-chip Multiplier and the Primary Accumulators are reset to zero. The maximum absolute correlation result that can be represented before wrap-around is 255, which is sufficient for a W-edge length up to 29 at a *CoreClk* frequency of 30 MHz and a nominal GPS P-code rate even for the case of a noise-free input signal.

W-chip Multiplier and Secondary Accumulator

At the end of each W-code chip, the 22-bit Secondary Accumulator subsequently adds the results from the Primary Accumulator multiplied with the estimated W-code chip from the Semi-codeless Controller. This addition is synchronised so that the W-code chip estimate originates from the same period as the results from the Primary Accumulator. At the end of each Integration Epoch the Secondary Accumulator values are latched into the six correlation registers *CorrValue0*I, *CorrValue0*Q, *CorrValue1*I, *CorrValue1*Q, *CorrValue2I* and *CorrValue2*Q, and the Primary and Secondary Accumulators are reset to zero.

Note: The last W-code chip in an Integration Epoch will be integrated in the next correlation values, instead of in the current ones. This will have an impact only when there is a data bit change, and the corresponding implementation loss is therefore insignificant.

W-edge Delay

The W-edge signal is delayed to allow it to be aligned with the corresponding E, *P* and *L* P-code sequences provided from the Code Delay Line. The W-edge signal is delayed by two *CoreClk* cycles for the *E* CaP-integrator, with three *CoreClk* cycles for the *P* CaP-integrator and with four *CoreClk* cycles for the *L* CaP-integrator.

Note: For best results the Code Delay Line should then be programmed with the same delays as specified here for the W-edges, and the CaP-integrators should operate with separate E, *P* and *L* code sequences. This is obtained by setting *CodeInSpace* to three, *ChipSpacingEP* and *ChipSpacingPL* to one in the *CorrMode* registers.

P-code Unit with Semi-codeless Controller

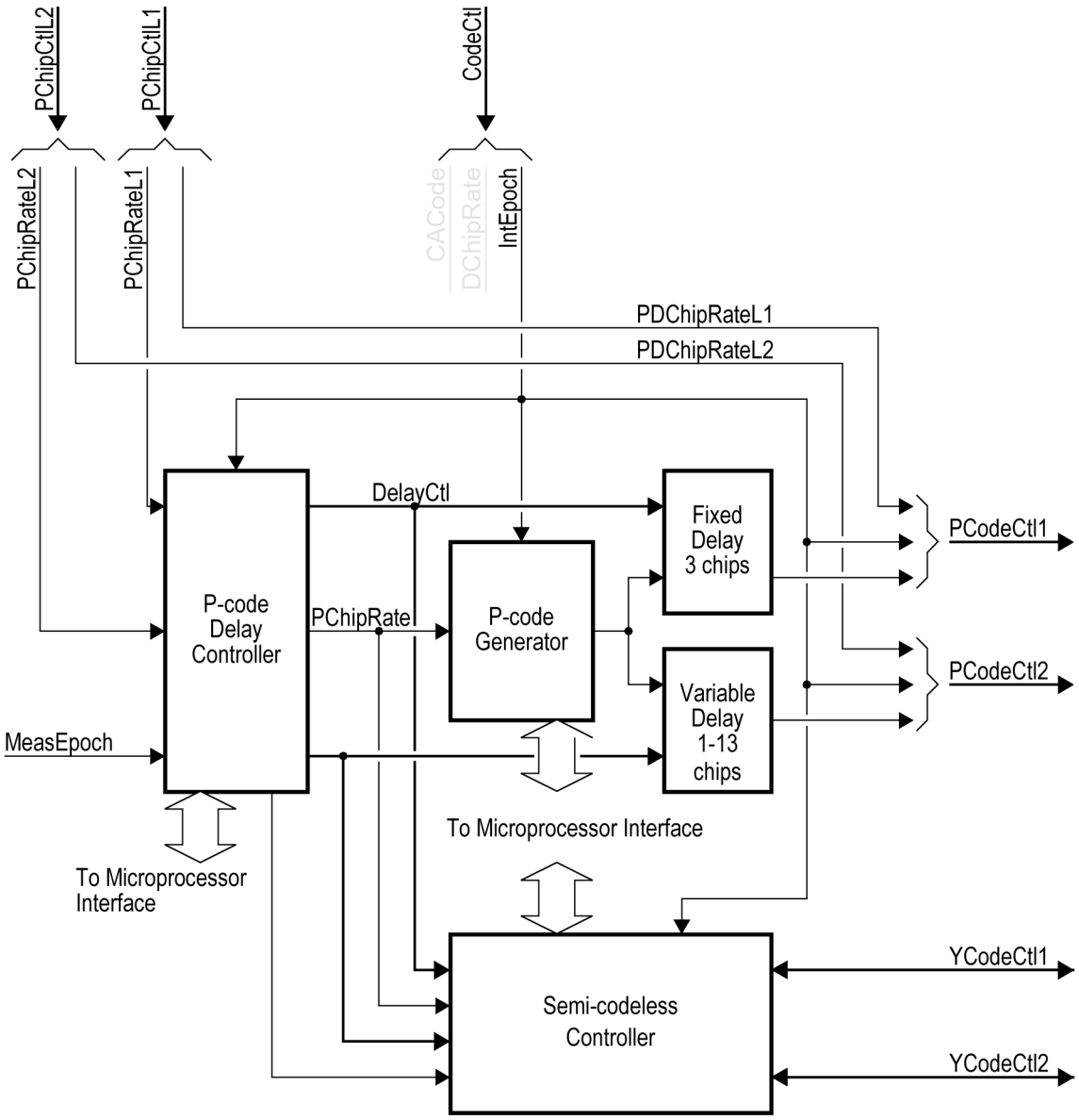
One P-code Unit is provided per dual-frequency channel. As can be seen in Figure 11, the P-code Unit replaces the function of the C/A-code Unit for the two CaP-channels. The P-code chip rates for the L1 and L2 signals are generated by the Code-rate Generators of the respective CaP-channels.

The P-code Unit shown in Figure 19 generates one replica GPS or GLONASS P-code sequence, a delayed version of this P-code sequence, and control signals for semi-codeless processing of GPS Y-code. It consists of the P-code Generator, the P-code Delay Controller, One Fixed Delay and one Variable Delay and the Semi-codeless Controller. The P-code Generator can produce a GPS or GLONASS replica P-code sequence, and includes functionality to initialise GPS P-code generation since the GPS P-code sequence is very long. The P-code sequence intended for the L1 frequency has a fixed delay of 3 P-code chips, while there is a Variable Delay for the P-code sequence intended for the L2 frequency can vary between 1 and 13 P-code chips, controlled by the P-code Delay Controller. The two P-code sequences are subsequently provided to the two CaP-channels of a dual-frequency channel. The Semi-codeless Controller generates control signals for the CaP-integrators for semi-codeless processing of Y-code *(YCodeCtl1* and *YCodeCtl*2), as described in page 38.

The number of L1 P-code chips and the delay difference between the P-code for L1 and L2 are sampled at the end of each Measurement Epoch so it can be read by the microprocessor. This observable can be used to obtain a high-precision pseudorange and to calculate the difference in delay between the L1 and L2 frequencies.

The P-code and Control *(PCodeCtl1* and *PCodeCtl*2) signals are provided to the CaP-channels, allowing processing of the P-code modulated on the L1 and L2 frequencies. Each *PCodeCtl* signal consists of a P-code sequence, the Integration Epoch provided by the CA-channel so that all Correlator Units of a dual-frequency channel integrate over the same period, and the corresponding double chip-rate signal *PDChiprate* for clocking the Code Delay Line of the CaP-channel. As can be seen in Figure 19, the *CACode* and *DChipRate* signals from the CA-channel are not used, being replaced by the signals corresponding to the two P-code sequences.

Figure 19. P-code Unit block diagram



P-code Generator in GLONASS Mode

The GLONASS P-code sequence has a length of 5’110’000 chips. It has a chip rate of 5.11 MHz and a code epoch of exactly one second. The shortened code sequence is taken from the tenth tap of a 25-stage LFSR with the generating polynomial:

$$G_P = 1 + X^3 + X^{25}$$

Generation of the replica GLONASS P-code sequence is programmed by setting the *SVSel* field of the *PCodeSetting* register to zero. At the same time the P-code Generator is stopped and reset to its initial state (all ones). This state corresponds to the one second mark in the GLONASS navigation message.

P-code Generator in GPS Mode

The GPS P-code sequence for one Space Vehicle has a length of 6’187’104’000’000 chips. It has a chip rate of 10.23 MHz and a code epoch of exactly one week, as defined in RD1. The code sequence is generated by the modulo-2 addition of the X1 and X2 sequences defined below. The code for Space Vehicle *i* is obtained by delaying the X2 sequence by *i* chips prior to the modulo-2 addition, where *i* is a value ranging from 1 to 37.

Generation of the replica GPS P-code is programmed by setting the *SVSel* field of the *PCodeSetting* register to the desired Space Vehicle number. At the same time the P-code Generator is stopped and initialised to the state determined by the *ZCount* field as described in page 37.

X1 Generator

The GPS X1 sequence has a period of 15’345’000 chips, which corresponds to exactly 1.5 second. It is generated by the modulo-2 addition of the outputs of two 12-stage LFSRs with the generating polynomials:

$$X1A = 1 + X^6 + X^8 + X^{11} + X^{12}$$

$$X1B = X^1 + X^2 + X^5 + X^8 + X^9 + X^{10} + X^{11} + X^{12}$$

The X1A sequence is shortened to the length 4092, while the X1B sequence is shortened to the length 4093. The X1 sequence is shortened to a length of 3750 X1A periods (15’345’000 chips) by halting the X1B LFSR in its final state when it has completed its 3749th period. After a period of 343 chips the X1A LFSR will complete its 3750th period, after which both the X1A and X1B LFSRs are reset to their initial states as defined in Table 3. This event also indicates the beginning of a new X1A epoch.

X2 Generator

The GPS X2 sequence has a period of 15’345’037 chips, which is 37 chips longer than the X1 epoch. It is generated by the modulo-2 addition of the outputs of two 12-stage LFSRs with the generating polynomials:

$$X2A = 1 + X^1 + X^3 + X^4 + X^5 + X^7 + X^8 + X^9 + X^{10} + X^{11} + X^{12}$$

$$X2B = 1 + X^2 + X^3 + X^4 + X^8 + X^9 + X^{12}$$

The X2A sequence is shortened to the length 4092, while the X2B sequence is shortened to the length 4093. The X2 sequence is shortened to a length of 3750 X2A periods plus

37 chips by halting the X2B LFSR in its final state when it has completed its 3749th period. After a period of 343 chips the X2A LFSR will complete its 3750th period, after which it is halted in its final state. After an additional 37 chips both the X2A and X2B LFSRs are reset to their initial states as defined Table 3**.**

Table 3. LFSR Initial states

LFSR	Initial State
X1A	001001001000
X1B	010101010100
X2A	100100100101
X2B	010101010100

GPS Z Counter

The 19-bit Z counter counts the number of elapsed X1 epochs since the start of the week, referred to as the Z count. It has a range of 0 to 403'199 X1 epochs (i.e., a period of 403'200 X1 epochs), where the final value represents End Of Week. During the last X1A period of a week the X1B, X2A and X2B generators will be halted in their final state. When the X1A generator subsequently reaches its final state all four LFSRs are reset to their initial states and the Z-count is reset to zero.

P-code Generator Initialisation for GPS

Writing a GPS P-code setting to the *PCodeSetting* register starts an initialisation of the P-code Generator. It takes (15'345'000 + Z-count) P-code chips to complete and once started it can not be interrupted. The first P-code chip generated after completed initialisation and handover will be the first chip of the X1 epoch corresponding to a Z-count of the programmed value + 1. Programming the maximum value 403199 will lead to the P-code generator starting at Z-count 0.

P-code Handover

The P-code Handover mechanism allows the firmware to start the P-code Generator at the precise moment corresponding to the state to which is has been initialised. The P-code Generator is started at the start of the first Integration Epoch after a write access to the *HandOver* register. P-code Handover applies to both GPS and GLONASS operation.

Note: If handover is performed before the initialisation for GPS has been started or has been completed, the handover might lead to the P-code Generator being started at some Integration Epoch with an undefined code being generated.

P-code Delay Controller

The same P-code sequence is used for the signals modulated on the L1 and L2 frequencies for one GPS satellite. However, due to different ionospheric and front-end group delays the two signals will have a relative delay at the *ADCIn* inputs of the AGGA-2. The maximum supported ionospheric group delay is 800 ns with L2 lagging L1. The maximum supported front-end group delay is 200 ns. As a consequence, the total delay between L1 and L2, and thus between the corresponding P-code sequences, can vary between -200 ns and 1 ∝s. For reference, one GPS P-code chip corresponds to approximately 100 ns, while one GLONASS P-code chip corresponds to approximately 200 ns.

One single P-code Generator together with two delays and the P-code Delay Controller are used to produce the two P-code sequences for the L1 and L2 signals with a resolution of one *CoreClk* cycle. The P-code sequence intended for L1 is always 3-P-code chips, whereas the P-code sequence intended for L2 is variable and controlled by the Cycle Difference Counter in the P-code Delay Controller. The Variable Delay is adjusted for a delay ranging from 1 to 13 P-code chips to obtain the desired delay between the P-code sequences.

The initial delay difference between the L1 and L2 sequences, expressed as an integer number of P-code chips, has to be calculated by the firmware and programmed by the *L1LagsL2* field and the *CycleDiffCnt* field of the *CycleDiff* register. The Cycle Difference Counter is updated with a newly programmed value in the *CycleDiff* register at the end of the next Integration Epoch. The update is only performed once after programming.

Subsequently, the instantaneous delay difference is indirectly controlled by the phase of the two CaP-channel Code-rate Generators (see page 23) as provided by the chip-rate signals. For each new L1 P-code chip the counter is incremented, while for each new L2 P-code chip it is decremented. If it is attempted to exceed the valid Cycle Difference Counter end values of −2 and +10 P-code chips, the Cycle Difference Counter will remain at the end value until reprogrammed by the *CycleDiff* register.

P-code Delays

The Delay for the P-code sequence intended for L1 (channels 1, 4, 7 and 10) is fixed to 3 P-code chips. The Variable Delay can delay the P-code sequence intended for L2 (channels 2, 5, 8 and 11) from 1 to 13 P-code chips with a resolution of one *CoreClk* cycle, according to the value of the Cycle Difference Counter in the P-code Delay Controller.

P-code Observable

At the end of each Measurement Epoch the instantaneous values of the Cycle Difference Counter *CycleDiffObs* and the P-code Chip Count *PChipCount* are sampled and placed in the P-code Observable register *PCodeObs*. The *CycleDiffObs* field contains a 5-bit value in sign/magnitude format with a valid range from -2 to +10, where a positive value indicates that the L2 signal lags the L1 signal.

The P-code Chip Count is obtained from a 24-bit counter counting the number of L1 P-code chips as signalled by the *PChipCtlL1* signal. The counter is reset at P-code Handover and when it has reached a count equal to the number of P-code chips in one second (5'110'000 P-code chips for GLONASS, 10'230'000 P-code chips for GPS).

The Cycle Difference Counter value represents the integer part of the accumulated phase difference between the L1 and L2 P-code sequences. The exact L1-L2 delay can be calculated as the effective sum of the *CycleDiffObs* value as integer part and the difference between the *CodePhase* fields of the *CodeObs* registers for the L1 and L2 CaP-channels as fractional part.

Semi-codeless Controller

The Semi-codeless Controller shown in Figure 9 generates one W-edge sequence and a delayed version of this W-edge sequence, and provides estimates of the W-code chips. It consists of the W-edge Generator, one Fixed Delay and one Variable Delay and two W-chip Estimators. The W-edge Generator produces a W-edge sequence which indicates the assumed limits of the W-code chips. The Fixed Delay and the Variable Delay subsequently delays this sequence, controlled by the P-code Delay Controller, so that the two resulting W-edge sequences are in phase with the corresponding P-code sequences. Each W-chip Estimator predicts the value of the W-code chip based on the primary correlation value in the Punctual CaP-integrator of the other channel. The primary correlation value is then multiplied by this estimated W-code chip to decorrelate with the estimated W-code sequence.

By virtue of an adjustable threshold, when the estimation is indecisive (i.e. the value from which to estimate the W-code chip is close to zero) the secondary accumulation is inhibited by setting the W-chip estimate to zero. As a result the W-chip signal can have one of the values [-1, 0, 1]. The number of estimations resulting in a decisive W-code chip is counted during one Integration Epoch. This number can then be used for adaptive control of the threshold levels, which are programmable by the microprocessor.

As part of the Y-code and Control *(YCodeCtl)* signal, one W-edge sequence *(WEdge)* together with one W-chip estimate *(WChip)* are provided to each of the two CaP-integrators. In addition, one primary correlation value from the Punctual CaP-integrator is provided to be used for estimating the W-chip for the other channel.

W-edge Generator

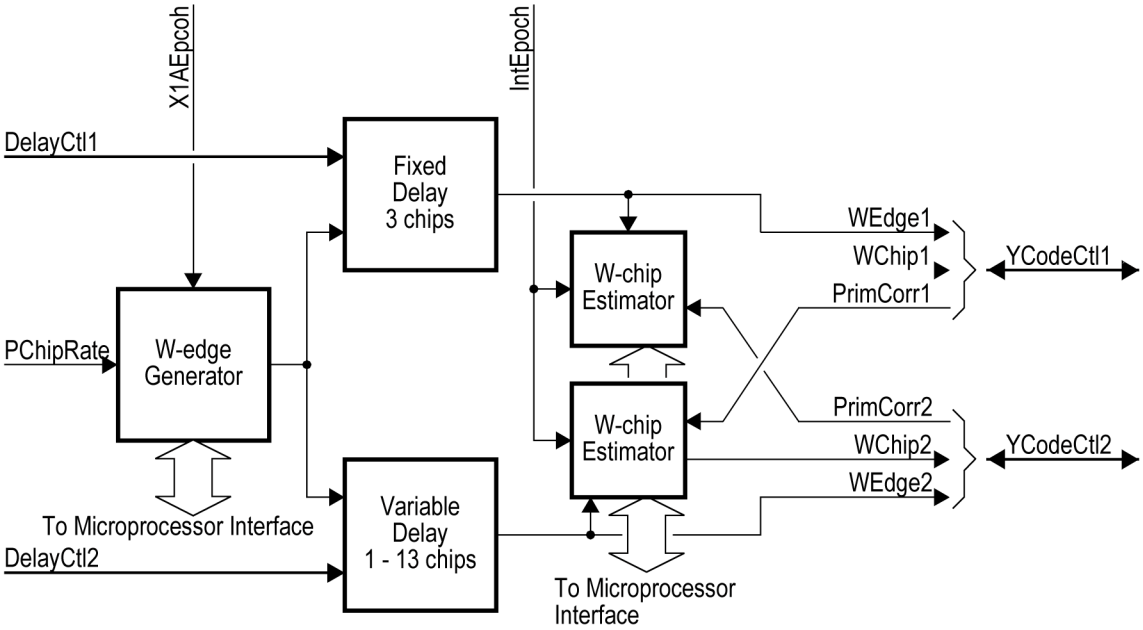
The W-edge Generator produces a W-edge sequence indicating the assumed limits of the W-code chips. It is controlled by the *PChipRate* signal from the P-code Delay Controller, which signals each new P-code chip. A W-edge is signalled after an integer number of P-code chips, as determined by programming the *A* and *B* parameters. To allow the generation of non-uniform W-edge sequences, the *A* parameter determines the length of the W-code for *M* W-code chips, after which the *B* parameter is applied for *N* W-code chips. This behaviour is repeated until the end of the current X1A epoch as indicated by the P-code generator. At the start of each new X1A epoch the W-edge gen-

erator is suspended for a number of P-code chips determined by the *S* parameter, after which the W-edge sequence generation is started using the *A* parameter.

The A, B, N, *M* and *S* fields of the *WEdgeCtl* register configures the W-edge Generator. The *A* and *B* parameters each have a valid range from 11 to 31, where the number of P-code chips for one W-code chip is the same as the programmed value. The *M* and *N* parameters can each have a value between 0 and 63 W-code chips. The Semi-codeless Controller including the W-edge Generator is disabled when *M* is zero. The *S* parameter can have a value between 0 and 31 P-code chips.

The A, B, N, *M* and *S* fields of the *WEdgeCtl* register configures the W-edge Generator. The *A* and *B* parameters each have a valid range from 11 to 31, where the number of P-code chips for one W-code chip is the same as the programmed value. The *M* and *N* parameters can each have a value between 0 and 63 W-code chips. The Semi-codeless Controller including the W-edge Generator is disabled when *M* is zero. The *S* parameter can have a value between 0 and 31 P-code chips.

Figure 20. Semi-codeless Controller block diagram



W-edge Delays

The delays for the W-edge sequences behave as the P-code Delays described in page 38, i.e. the delay intended for L1 is fixed to 3 P-code chips while the delay intended for L2 is variable between 1 and 13 P-code chips, controlled by P-code Delay Controller. These accept a W-edge sequence from the W-edge Generator and provide the delayed W-edge sequences to the W-chip Estimators and to the CaP-Integrators of both Cap-integrators in the dual-frequency channel.

W-chip Estimator

The W-chip Estimator of the L1 CaP-channel compares the L2 primary correlation value *PrimCorr2* with the threshold level programmed by the *Threshold1* field of the *Threshold* register. When the absolute *PrimCorr2* value is lower than or equal to the threshold level, *WChip1* is set to zero so that the L1 primary correlation values are not accumulated.

Otherwise, *WChip1* is set to the same sign as *PrimCorr*2, using the value -1 or 1. *WChip1* is synchronised using the *WEdge1* signal so that it is multiplied with the primary correlation values originating from the same W-code chip as used for the estimation.

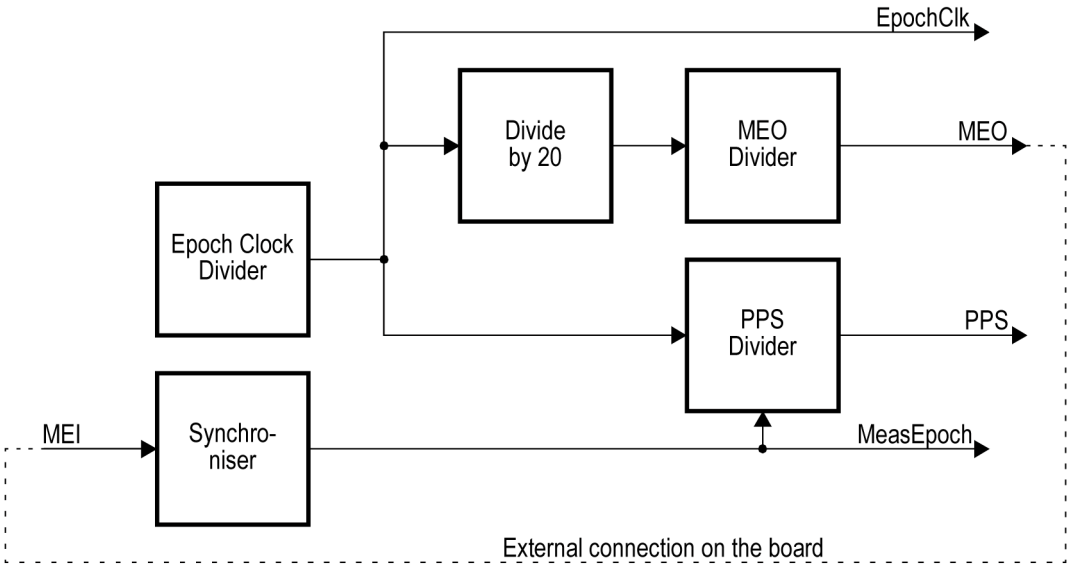
Every time a decisive W-code chip is estimated, i.e. the primary correlation values are accumulated, a counter is incremented. At the end of each Integration Epoch this counter value is latched into the *Decisive1* field of the *Decisive* register.

The W-chip Estimator for the L2 CaP-channel operates in the corresponding way, using the L1 primary correlation value *PrimCorr*1, the *Threshold2* and *Decisive2* fields, and the *WChip2* and *WEdge2* signals.

Time Base Generator

This section describes the Time Base Generator. As shown in Figure 20 it consists of the Epoch Clock Divider generating the *EpochClk* clock, a divide-by-20 divider, the MEO Divider generating the Measurement Epoch Output *(MEO)*, the PPS Divider generating the Pulse-Per-Second *(PPS)* strobe and a Synchroniser for the Measurement Epoch Input *(MEI)* providing the internally used Measurement Epoch strobe *(MeasEpoch)*.

Figure 21. Time Base Generator block diagram



EpochClk can have a frequency up to 6 MHz, with a typical value being 1 kHz. It is used to generate the Measurement Epoch and the Pulse-Per-Second Strobe. It is also used in the Signal Level Detector in the Front-end Interface and in the Antenna Switch Controller.

MEO is generated from *EpochCl*k, and can have a frequency up to *CoreCl*k/129. A typical value is 10 Hz.

The *MeasEpoch* strobe signals the end of each Measurement Epoch and is used in the Channel Matrix to sample the observables. An interrupt can be generated to inform the microprocessor that new values are available. *MeasEpoch* is taken from the Measurement Epoch Input *(ME*I) after synchronisation and is used in the Channel Matrix.

In receivers with only one AGGA-2, *MEI* should be externally connected to its own *MEO* output. In receivers with multiple AGGA-2s, all *MEI* inputs should be connected to the *MEO* output from one AGGA-2, referred to as the master AGGA-2, to ensure that all channels in the receiver use the same Measurement Epoch. However, since *MEO* has a known timing relationship with the receiver time it is in principle possible for two stand-alone receivers to adjust their Measurement Epochs with an accuracy better than 1 ∝s.

The *PPS* output is intended for synchronising external equipment to the receiver time. *PPS* is derived from *EpochClk* and can have a frequency up to *CoreCl*k/129. A typical value is one pulse per second or one pulse per ten seconds. An interrupt can be generated to inform the microprocessor that the *PPS* output has been asserted.

Epoch Clock

The Epoch Clock Divider generates the *EpochClk* internal clock by dividing the *CoreClk* clock by a fixed-point value. The valid frequency range is from *CoreCl*k/65534 to *CoreClk*/5, corresponding to the range 460 Hz to 6 MHz for a *CoreClk* frequency of 30 MHz. The resolution is approximately $f_e^2/f_c/1024$, where f_e is the *EpochClk* frequency and f_c is the *CoreClk* frequency, corresponding to 33 ∝Hz for an *EpochClk* frequency of 1 kHz and a *CoreClk* frequency of 30 MHz.

The *EpochClk* frequency f_e is obtained by programming the *EpochClkDiv* register with the value $f_c/f_e - 1$ expressed as a 26-bit fixed-point value having a 16-bit integer part and a 10-bit fractional part, where f_c is the *CoreClk* frequency. A division ratio lower than 5 is not supported. A newly programmed value becomes effective at the start of the next *EpochClk* cycle.

Note: For non-integer division ratios the number of *CoreClk* cycles per *EpochClk* can differ by one *CoreClk* cycle for different *EpochClk* cycles.

Measurement Epoch

The *MeasEpoch* strobe signals the end of each Measurement Epoch. It is taken from the *MEI* input after synchronisation and detection of the rising edge. An interrupt can be generated for each new Measurement Epoch, as described in page 46.

MEO is obtained by first dividing the *EpochClk* clock by a fixed ratio of 20 and then dividing it further by a programmable division ratio ranging from 1 to 64. The division ratio is programmed by the *MEODivRatio* field of the *TimeBaseSetting* register, and a new value becomes effective at the start of the next *MEO* cycle. The *MEOEnable* field controls whether the *MEO* output is enabled or constantly deasserted. When enabled, the *MEO* output is asserted for 128 *CoreClk* cycles at the beginning of each *MEO* period.

The number of *CoreClk* cycles since the last Measurement Epoch is counted modulo 65536 by a 16-bit wrap-around counter. At the end of each Measurement Epoch this count is sampled into the *MEPeriod* field of the *PPSMEState* register. The *MEPeriod* value can be used by the firmware to determine the exact number of *CoreClk* cycles between two Measurement Epochs if the approximate number is known.

Pulse-Per-Second

The *PPS* output is intended for synchronising external equipment to the receiver time. *PPS* is derived from the *EpochClk* clock by division with a programmable division ratio. The *PPS* strobe can be synchronised to a reference time as described in page 42. *PPS* is obtained by dividing the *EpochClk* clock by a programmable division ratio ranging from 1 to 16384. The division ratio is programmed by the *PPSDivRatio* field of the *TimeBaseSetting* register. An interrupt can be generated when *PPS* has been asserted, as described in page 46. The *PPSEnable* field controls whether the *PPS* output is enabled or constantly deasserted. It does not affect the operation of the PPS Divider or the interrupt, which allows the *PPS* signal to be synchronised to a reference time before being activated externally. When enabled, the *PPS* output is asserted for 128 *CoreClk* cycles at the beginning of every *PPS* cycle. The delay from *EpochClk* to the *PPS* output is one *CoreClk* cycle shorter than the delay from *EpochClk* to *MeasEpoch* (assuming a direct connection from *MEO* to *ME*I), so that the rising edge of *PPS* occurs one *CoreClk* cycle before the sampling of the observables at the end of the Measurement Epoch. A new value for the *PPSDivRatio* field becomes effective at the start of the next *PPS* cycle.

Synchronisation to a Reference Time Signal

The *PPS* strobe can be synchronised to a reference time such as UTC time, GPS or GLONASS system time with a precision of one *CoreClk* cycle at an *EpochClk* frequency of 1 kHz. At the end of every Measurement Epoch the state of the *PPS* divider is stored in the *PPSDivState* field of the *PPSMEState* register. This value can be read by the microprocessor, to determine the remaining number of *EpochClk* cycles until the *PPS* strobe will be asserted. Since the firmware calculates the local clock bias for the Mea-

surement Epoch as part of the navigation solution, the total time offset for the *PPS* strobe w.r.t. a selected reference time can be established. Synchronisation can be obtained by first adjusting the *PPS* division ratio to obtain an accuracy within one *EpochClk* cycle for the *PPS* strobe. Optionally, the Measurement Epoch could also be adjusted to coincide with the *PPS* strobe. Thereafter the *EpochClk* phase is adjusted to be exactly in phase with the selected reference time by reprogramming the *EpochClkDiv* register. When the clock bias w.r.t. *EpochClk* has been eliminated from the navigation solution, the *PPS* strobe is synchronised to the selected reference time.

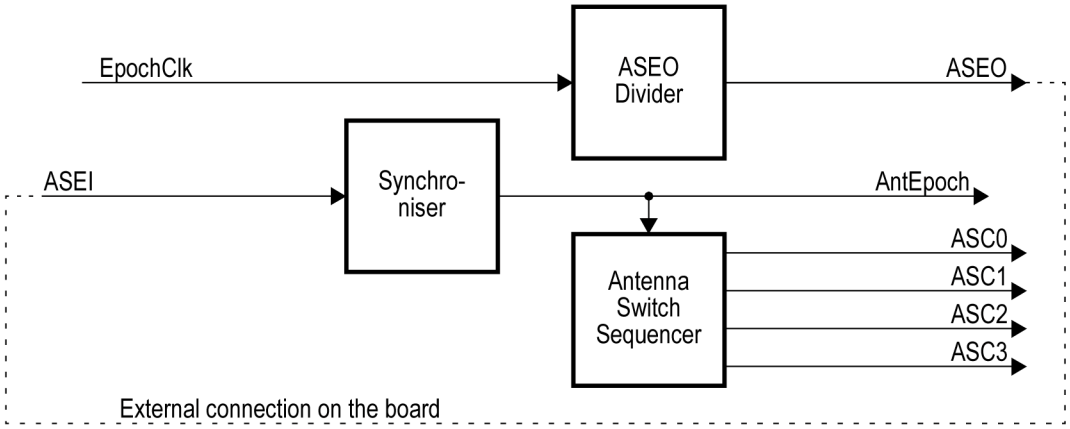
Note: When the *EpochClk* phase is adjusted this will affect the length of the Measurement Epoch and the Signal Level Detector integration interval, since they are derived from *EpochCl*k. Depending on the application, compensation by the firmware might be needed.

Note: If the GNSS receiver is used in an application using elapsed time rather than an earth-based reference time, the division ratios for *EpochClk* and *PPS* can be programmed to a fixed and integer division ratio. The navigation solution can then be used to establish the time bias for the elapsed time, but without synchronising the *PPS* output.

ANTENNA SWITCH CONTROLLER

This section describes the Antenna Switch Controller. As shown in Figure 21, it consists of the ASEO Divider, a Synchroniser for the Antenna Switch Epoch Input *ASEI* and the Antenna Switch Sequencer.

Figure 22. Antenna Switch Controller block diagram



The Antenna Switch Controller supports the Hybrid Parallel-Multiplex attitude determination scheme (RD5). The four Antenna Switch Control *(ASC)* outputs are intended for controlling external antenna switching of up to four antennas. It also generates the Antenna Switch Epoch *(AntEpoc*h) strobe which signals antenna switch events, and which is used to control the updating of the correlation values for the slaved channels. An interrupt can be generated for each Antenna Switch Epoch. Both the *AntEpoch* strobe and the *ASC* outputs are generated from the Antenna Switch Epoch Input *(ASE*I).

In receivers with only one AGGA-2 *ASEI* should be externally connected to its own *ASEO* output. In receivers with multiple AGGA-2s all *ASEI* inputs should be connected to the *ASEO* signal from one AGGA-2, referred to as the master AGGA-2, to ensure that all channels in the receiver use the same Antenna Switch Epoch. The Antenna Switch Epoch Output *(ASE*O) is generated from the *EpochClk* clock, and can have a frequency up to *CoreCl*k/129. A typical value is
200 Hz.

Note: A patent has been granted for the Hybrid Parallel-Multiplex attitude determination technique. Users intending to use this technique are advised to contact the Patent Office at ESA Headquarters for further information.

Antenna Switch Epoch

The *AntEpoch* strobe signals the end of each Antenna Switch Epoch. It is taken from the *ASEI* input after synchronisation and detection of the rising edge. An interrupt can be generated for each Antenna Switch Epoch, as further described in page 46.

ASEO is obtained by dividing the *EpochClk* clock by a programmable division ratio ranging from 1 to 32. The division ratio is programmed by the *ASEODivRatio* field of the *AntSwitchMode* register. The *ASEOEnable* controls whether the *ASEO* output is enabled or constantly deasserted. When it is disabled the ASEO Divider is also disabled. The *ASEO* output is asserted for 128 *CoreClk* cycles.

Antenna Switch Sequencer

The Antenna Switch Sequencer generates the four Antenna Switch Control outputs *ASC*0, *ASC*1, *ASC2* and *ASC*3. These four outputs are intended for controlling the external antenna switching of up to four antennas to a common RF/IF section.

At the start of each Antenna Switch Epoch the Antenna Switch Sequencer asserts the next higher *ASC* output that has been enabled by the *ASCEnable0* to *ASCEnable3* bits of the *AntSwitchMode* register. In this way the enabled *ASC* outputs are activated in sequence. Only one *ASC* output is asserted at any time, and it remains asserted until the next one is asserted. If no *ASC* output has been enabled the sequencer is disabled and all *ASC* outputs are deasserted.

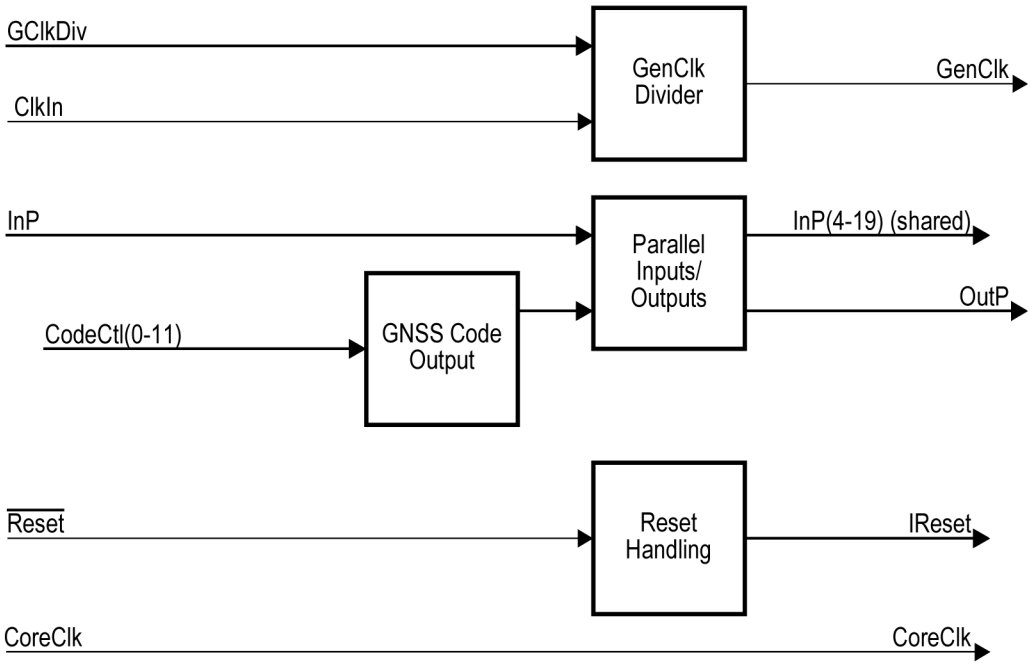
At the end of each Antenna Switch Epoch the number of the *ASC* output that was asserted is stored in the read-only *AntSwitchId* field of the *AntSwitchMode* register. By reading this value the firmware can establish to which antenna the correlation values correspond.

Note: To reduce the impact of switching noise the accumulation in the Integrators is inhibited at the start of each Antenna Switch Epoch, as described in page 32.

SYSTEM SUPPORT FUNCTIONS

This section describes the System Support Functions. As shown in Figure 22 this includes the generation of an external clock, a basic parallel Input/Output interface, GNSS code output and reset handling.

Figure 23. System Support Functions block diagram



The GenClk Divider produces the General Purpose Clock *(GenCl*k). This independent clock is derived from the *ClkIn* clock input and can for example be used for clocking the *CoreClk* input, the GP2010 front-end (RD10), the receiver micro-processor, etc. The clock frequency is determined by the 3-bit *GClkDiv* input.

The parallel Input/Output interface consists of the 20-bit Input Port *(In*P) and the 12-bit Output Port *(Out*P). They can be used for general functions such as monitoring lock indicators of RF down-converters and power-down of external circuitry. In addition, 16 of the *InP* inputs are shared with the Front-end Interface to support redundancy for complex inputs, as described in page 8. Two of the outputs can be configured to provide the code sequence and the corresponding Integration Epoch of any channel, which can be used for GNSS transmission, calibration or debugging.

The AGGA-2 is reset when the *Reset* input is asserted or by a microprocessor write access to the *ChipReset* register. The Internal Reset *(IReset*) signal consists of multiple different reset signals, and is connected to all blocks in the AGGA-2.

The *CoreClk* clock input provides the AGGA-2 internal clock which is provided to all blocks in the AGGA-2.

Clock Generation

GenClk

GenClk is intended for general usage, such as providing the clock for the microprocessor or the *CoreClk* input clock. It is generated by dividing *ClkIn* by a factor of two to eight, determined by the 3-bit *GClkDiv* input as specified in page 69. Alternatively, the *GenClk* divider can be disabled. When *ClkIn* is divided by an even number *GenClk* is symmetric. When *ClkIn* is divided by an odd number the duty cycle is not 50%, the clock output it is high for one more *ClkIn* period than it is low.

Reset Handling

When the *Reset* signal is asserted the entire AGGA-2 is reset, and all registers and signals are to their initial states as specified in page 72. When reset, most internal functions are disabled to reduce power consumption.

The *ChipReset* register is provided to allow recovery from unexpected events such as Single Event Upsets. Writing to this register will reset the AGGA-2 with exception of the Epoch Clock Divider, the MEO Divider and the PPS Divider in the Time Base Generator, and the GenClk divider. By not resetting these functions the impact on the *PPS* output and the *GenClk* clock can be minimised for a reset by the *ChipReset* register. It can take up to five *CoreClk* cycles before a reset is recognised. After a reset, the AGGA-2 registers are initialised which takes 32 *CoreClk* cycles, during which they should not be accessed.

Input and Output Ports

The parallel Input/Output interface consists of the 20-bit Input Port *(In*P) and the 12-bit Output Port *(Out*P). The value of the *InP19 0* inputs can be read from the *InP* register. The *OutP11 0* outputs have the same value as the corresponding bits of the *OutP* register.

Note: Since intended for slowly varying signals, the signals for the Input Port are not latched when accessed, and the value read from the *InP* register can change during the access. Be reading the *InP* register until the two last values read are identical a coherent value for the entire register can be obtained.

GNSS Code Output

The *OutP11* and *OutP10* outputs can be configured to provide the code sequence and the corresponding Integration Epoch of any channel, which can be used for debugging, calibration of the RF front-end and generation of GNSS signals for transmission, e.g. in test equipment or for distance measurement systems.

Whether the *OutP11* and *OutP10* outputs get their value from the *OutP* register or provide a code sequence with associated Integration Epoch indication is determined by the *CodeEnable* bit of the *CodeOutput* register. When enabled, the *Channel* field selects from which channel the code sequence shall be output, as provided by the Code and Control signal *(CodeCt*l). *OutP11* then provides the Integration Epoch from the selected channel, delayed by two *CoreClk* cycles. At the end/beginning of each Integration Epoch this signal is logical zero for one *CoreClk* cycle, and then remains at logical one during the remaining Integration Epoch. *OutP10* provides the code sequence from the selected channel, taken from the *CodeOut* signal shown in Figure 15 on page 28. The code sequence is delayed by 3 to 18 *CoreClk* cycles, as programmed by the *CodeDelay* field. The relationship between the output GNSS code sequence and the E, *P* and *L* code sequences used for correlation can be further adjusted by the Code Delay Line described in page 26.

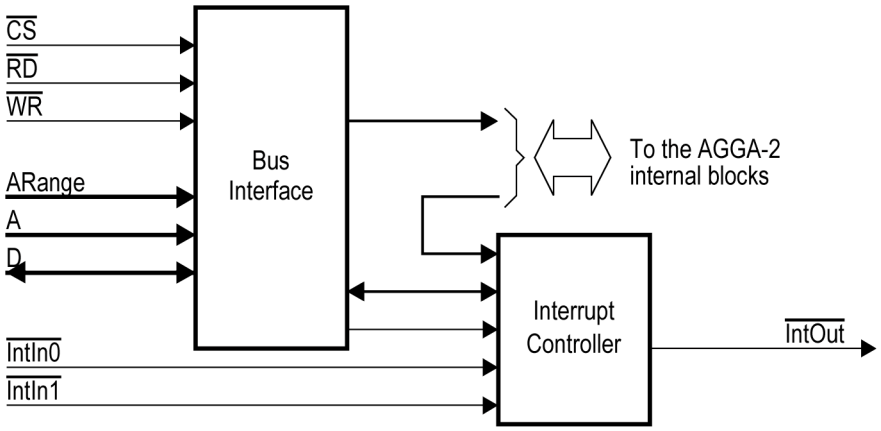
Note: For proper operation *CodeDelay* shall correspond to less than one code chip.

Note: For proper operation at high *CoreClk* frequencies the code sequence should not be selected from a slaved channel.

MICROPROCESSOR INTERFACE

This section describes the Microprocessor Interface. As shown in Figure 23 it consists of the Bus Interface and the Interrupt Controller.

Figure 24. Microprocessor Interface block diagram



A microprocessor can access the AGGA-2 through the microprocessor interface as a generic memory-mapped peripheral. The interface consists of the 10-bit address bus *(*A), the 32-bit data bus *(*D), and the Chip Select *(C*S), Read *(R*D) and Write *(W*R) strobes. It is designed to be directly compatible with both the ADSP21020 DSP microprocessor (RD1) and the ERC32 Sparc chip set (RD7, RD8, RD9). By hardwiring the 2-bit *ARange* input, up to four different AGGA-2s can be accessed using a single chip select signal.

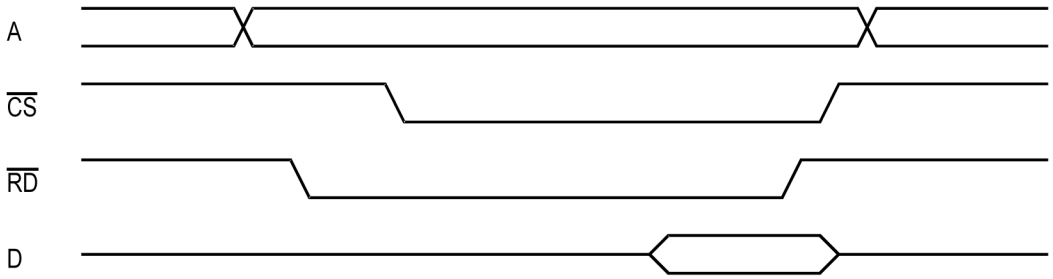
As correlation values, observables etc. become available in the AGGA-2, an external interrupt can be generated to inform the microprocessor that these can be read out. The Interrupt Controller stores the interrupt status and generates the external interrupt request output *IntOu*t. The AGGA-2 has two external interrupt inputs *IntIn0* and *IntIn*1, allowing multiple AGGA-2s to be connected in a tree configuration to a single interrupt input at the microprocessor.

Bus Interface

The *A9* and *A8* address bits are compared with the values on the two *ARange* inputs, and the AGGA-2 can only be accessed when these address bits matches the value on *ARange*. Address bit *A0* to *A7* decodes which register shall be accessed according to the register map defined in Table 4 on page 48. The 32-bit data bus *D* provides read and write access to the AGGA-2 registers.

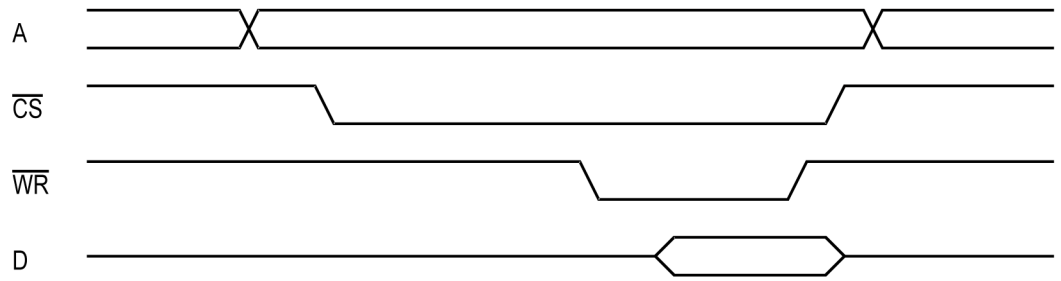
A read access to the AGGA-2 is shown in Figure 24. In this example, after the address has become stable and the *CS* and *RD* strobes have been asserted, data from the selected register is driven onto the *D* data bus by the AGGA-2. Due to the timing parameters the Bus Interface might require one or more wait-states for proper operation.

Figure 25. Example Read Access to the AGGA-2



An example write access to the AGGA-2 is shown in Figure 25. The *WR* strobe is gated so it does not become effective internally in the AGGA-2 until *CS* is asserted and the address matches the selection on *ARang*e. The data to be written is latched on the rising edge of *WR* while *CS* is asserted. After a write access to the AGGA-2 no new write access should be made for at least four *CoreClk* cycles to allow the value to allow sufficient time to properly synchronise the write access on the chip. A newly written value might not have the correct value if read during the five *CoreClk* cycles immediately following the end of the write access.

Figure 26. Example write access to the AGGA-2



Interrupt Controller

There are six types of interrupts that can be generated by the AGGA-2:

- Integration Epoch interrupts (one per channel);
- Measurement Epoch interrupt (one);
- Antenna Switch Epoch interrupt (one);
- Pulse-Per-Second interrupt (one);
- Signal Level Detector interrupt (one);
- Daisy-chain interrupts from other AGGA-2s or external devices (two).

There are in total 18 interrupt sources in the AGGA-2, all having equal priority. Each time an interrupt is signalled by the internal block that generates it, the interrupt status is updated in an internal shadow register, for each new interrupt setting the corresponding bit. The external interrupts corresponding to the *IntIn0* and *IntIn1* inputs are not stored in the shadow register. The interrupt request output *IntOut* is asserted when there is any unmasked interrupt represented either in the shadow register or on the external *IntIn* signals.

Each interrupt can be masked through the *IntMask* register. This does not affect the generation of interrupts internally in the AGGA-2; each interrupt is still registered in the internal status register. The external daisy-chain interrupts are neither masked while *Reset* is asserted nor after reset, to ensure these interrupts also propagate when the AGGA-2 is reset. While *Reset* is asserted the path is purely combinational to ensure the interrupts propagate even in the absence of the *CoreClk* clock. All other interrupts are masked during and after reset.

When the microprocessor has recognised the interrupt request on the *IntOut* signal, it should acknowledge it by performing a write access to the *IntStatus* register (with arbitrary data). Within eight *CoreClk* cycles after the end of the write access the contents of the internal shadow register have been transferred to the *IntStatus* register, and the internal shadow register is reset, ready to register any new internal interrupts. This also de-asserts the *IntOut* signal (unless any of the external interrupt inputs are asserted). The *IntStatus* register now contains the latched status of the internal registers, which will not change until the next interrupt acknowledge, while the for the interrupts corresponding to

the *IntIn0* and *IntIn1* inputs the corresponding bits in the *IntStatus* register will reflect the momentary values at these inputs. An active interrupt is represented by a one in the corresponding bit of the *IntStatus* register. All interrupts that have occurred are presented in the *IntStatus* register, regardless of whether masked or not.

Integration Epoch Interrupts

At the end of every Integration Epoch each channel generates an Integration Epoch interrupt which is registered by the corresponding *IntEpochInt* bit of the *IntStatus* register. These interrupts can be individually masked by the *IntEpochMask* field of the *IntMask* register. When a channel is disabled or slaved, its Integration Epoch interrupt should be masked.

Measurement Epoch Interrupt

At the end of each Measurement Epoch the Measurement Epoch interrupt is generated, which is registered by the *MeasEpochInt* bit of the *IntStatus* register. This interrupt can be masked by the *MeasEpochMask* field of the *IntMask* register. In a configuration with multiple AGGA-2s only one AGGA-2 should generate this interrupt.

Antenna Switch Epoch Interrupt

At the end of each Antenna Switch Epoch the Antenna Switch Epoch interrupt is generated, which is registered by the *AntEpochInt* bit of the *IntStatus* register. This interrupt can be masked by the *AntEpochMask* field of the *IntMask* register. In a configuration with multiple AGGA-2s only one AGGA-2 should generate this interrupt.

Pulse-Per-Second Interrupt

Each time the internal *PPS* strobe is asserted an interrupt is generated, which is registered by the *PPSInt* bit of the *IntStatus* register. The interrupt is generated regardless whether the *PPS* output is enabled or not. This interrupt can be masked by the *PPSMask* field of the *IntMask* register.

Signal Level Detector Interrupt

At the end of each Signal Level Detector integration interval an interrupt is generated, which is registered by the *LevelInt* bit of the *IntStatus* register. This interrupt can be masked by the *LevelMask* field of the *IntMask* register.

Daisy-Chain Interrupts

The *IntIn0* and *IntIn1* inputs allow multiple AGGA-2s to share a single interrupt input to the microprocessor. When unused these inputs should be tied to logical one. The *IntIn* field of the *IntStatus* register reflect the values at the *IntIn0* and *IntIn1* inputs at all times; reading the register will not reset this field. The daisy-chain interrupts can be masked by the *IntInMask* field of the *IntMask* register.

PROGRAMMING

This section specifies the register map, and defines all registers and their fields. Programming aspects for reducing the power consumption of unused functions and the programming restrictions are also described.

Register Map

The AGGA-2 register map has 256 possible addresses, shown as 17 register groups in Table 4. The first twelve groups correspond to the channels, the four subsequent groups to the P-code Units and the last group contains the common registers such as for the Time Base Generator, the Antenna Switch Controller and the Interrupt Controller. All addresses are given in hexadecimal notation.

Table 4. Overall AGGA-2 register map

Address	Register group
00 –0F	CA-channel 0
10 –1F	CaP-channel 1
20 –2F	CaP-channel 2
30 –3F	CA-channel 3
40 –4F	CaP-channel 4
50 –5F	CaP-channel 5
60 –6F	CA-channel 6
70 –7F	CaP-channel 7
80 –8F	CaP-channel 8
90 –9F	CA-channel 9
A0 –AF	CaP-channel 10
B0 –BF	CaP-channel 11
C0 –C7	P-code Unit 0
C8 –CF	P-code Unit 1
D0 –D7	P-code Unit 2
D8 –DF	P-code Unit 3
E0 –FF	Common registers

Each of the four P-code Unit register groups has eight registers as shown in Table 5.

Table 5. P-code Unit register group

Address	Register	Definition	Access
C0, C8, D0, D8	PCodeSetting	Table 23	R/W
C1, C9, D1, D9	CycleDiff	Table 25	R/W
C2, CA, D2, DA	PCodeObs	Table 26	R
C3, CB, D3, DB	HandOver	Table 24	W
C4, CC, D4, DC	WEdgeCtl	Table 27	R/W
C5, CD, D5, DD	Reserved		
C6, CE, D6, DE	Threshold	Table 28	R/W
C7, CF, D7, DF	Decisive	Table 29	R

Each of the twelve channel register groups has 16 registers as shown in Table 6.

Table 6. Channel register groups; valid range of “x” is 0 to B

Address	Register	Definition	Access
x0	ChannelMode	Table 11	R/W
x1	CarrierFreq	Table 12	R/W
x2	CarrierShift	Table 13	R/W
x3	CodeFreq	Table 14	R/W
x4	CodeShiftCtl	Table 15	R/W
x5	IntEpochLen	Table 16	R/W
x6	CodeSetting	Table 17	R/W
x7	CorrMode	Table 18	R/W
x8	CarrierObs	Table 19	R
x9	CodeObs	Table 20	R
xA	CorrValue0I	Table 21	R
xB	CorrValue0Q	Table 22	R
xC	CorrValue1I	Table 22	R
xD	CorrValue1Q	Table 22	R
xE	CorrValue2I	Table 22	R
xF	CorrValue2Q	Table 22	R

The common register group has 13 registers as shown in Table 7.

Table 7. Common register group

Address	Register	Definition	Access	Functional block
E0	ChipReset	Table 34	W	System Support Functions
E1	Reserved			
E2	IntMask	Table 38	R/W	Interrupt Controller
E3	IntStatus	Table 39	R/W	
E4 –E5	Reserved			
E6	AntSwitchMode	Table 33	R/W	Antenna Switch Controller
E7 – EA	Reserved			
EB	OutP	Table 36	R/W	System Support Functions
EC	InP	Table 35	R	
ED	CodeOutput	Table 37	R/W	
EE – EF	Reserved			
F0	TimeBaseSetting	Table 30	R/W	Time Base Generator
F1	EpochClkDiv	Table 31	R/W	
F2	Reserved			
F3	PPSMEState	Table 32	R	
F4 - F9	Reserved			
FA	InputMode	Table 8	R/W	Front-end Interface
FB	LevelIntCtl	Table 9	R/W	
FC	LevelValue	Table 10	R	
FD – FF	Reserved			

Front-end Interface Registers

InputMode (R/W)

The *InputMode* register shown in Table 8 consists of the four *ConvMode*3 *0* fields which determine the conversion mode of the Input Modules, and the *InputFormat* field which is used to select the input format, see page 9. A write to the *InputMode* register becomes effective immediately.

Table 8. InputMode Register

Bit	Field	Description
31-14	Reserved	
13-12	InputFormat	Selection of input format for the ADCIn inputs 00 = sign/magnitude format (default) 01 = unsigned format 10 = two's complement format 11 = comparator ladder format
11- 9	ConvMode3	Conversion mode for Input Module 3 000 = both inputs disabled (default) 001 = real mode, even input enabled, odd input disabled 010 = real mode, even input disabled, odd input enabled 011 = real mode, both inputs enabled 100– 110 = reserved 111 = complex mode
8- 6	ConvMode2	Idem for Input Module 2
5 -3	ConvMode1	Idem for Input Module 1
2-0	ConvMode0	Idem for Input Module 0

After reset the *InputMode* register contains all zeroes, disabling all *ADCIn* inputs and setting all *IQ* outputs to -1.

LevelIntCtl (R/W)

The *LevelIntCtl* register shown in Table 9 controls the Signal Level Detector, see page 11. The *IntLength* field specifies the integration interval equal to the programmed value times the *EpochClk* period. The *SignMagnSel* field determines whether positive samples or samples with an absolute value of 3 are counted. The *SignalSel* field selects which of the eight input signals is measured, and the *IQSel* bit selects either the I branch or the Q branch.

The *LevelIntCtl* register is double-buffered. When the Signal Level Detector is enabled, a newly programmed value becomes effective at the start of the next integration interval. When the Signal Level Detector is disabled *(IntLength* field is zero), a newly programmed value becomes effective immediately.

Table 9. LevelIntCtl register

Bit	Field	Description
31 -17	Reserved	
16-14	SignalSel	Select input signal 0 (default) to 7
13	IQSel	Select I or Q 0 = select I branch (default) 1 = select Q branch
12	SignMagnSel	Select the sign or magnitude bit 0 = count samples of absolute magnitude 3 (default) 1 = count positive samples
11- 8	Reserved	
7 -0	IntLength	Integration length 0 = disable the Signal Level Detector (default) 1 to 255 = integration length in number of EpochClk periods

After reset the *LevelIntCtl* register contains all zeroes, disabling the Signal Level Detector.

LevelValue(R)

The *LevelValue* register shown below provides the result from the Signal Level Detector, being the count value corresponding to the selected signal and signal characteristic (sign or magnitude), see page 11. The register is updated at the end of each integration interval. The *LevelVal* field contains the integrated value while bits 27 to 31 indicates the corresponding setting.

Table 10. LevelValue Register

Bit	Field	Description
31 - 29	SignalSel	Value of SignalSel corresponding to LevelVal
28	IQSel	Value of IQSel corresponding to LevelVal
27	SignMagnSel	Value of SignMagnSel corresponding to LevelVal
26 - 23	Reserved	
22 - 0	LevelVal	Value of measured signal

After reset and when the Signal Level Detector is disabled the *LevelValue* register has an undefined value.

Note: After being enabled, the length of the first integration interval might be up to one *EpochClk* period shorter than what was programmed by the *LevelIntCtl* register.

Channel Registers

ChannelMode (R/W)

The *ChannelMode* register shown in Table 11 determines the operating mode of the channel and the channel slaving, see page 16, page 20, page 25 and page 23. For a CA-channel bit 4 is a reserved bit, since no P-code can be selected. A write to the *ChannelMode* register becomes effective immediately.

Table 11. ChannelMode register

Bit	Field	Description
31 - 9	Reserved	
8 - 6	InputSel	Selection of input signal from the Front-end Interface 000 = select IQ0 (default) 001 = select IQ1 010 = select IQ2 011 = select IQ3 100 = select IQ4 101 = select IQ5 110 = select IQ6 111 = select IQ7
5	Reserved	
4 - 3	CodeSel	Replica code sequence & Control signal selection 00 = select internal C/A-code & Control signals (default) 01 = select Code & Control signals from channel c-1 10 = reserved 11 = select P-code & Control signals (CaP channel only)
2	CarrierSel	Replica carrier selection 0 = select carrier from internal Carrier Generator (default) 1 = select carrier from channel c-1
1	Reserved	
0	ChannelOn	Channel enabling 0 = disable channel and power-down (default) 1 = enable channel

After reset the *ChannelMode* register contains all zeroes, disabling the channel.

CarrierFreq (R/W)

The *CarrierFreq* register shown in Table 12 determines the carrier frequency, see page 21. To obtain the frequency f_{ca} ($-CoreClk/2 \leq f_{ca} \leq CoreClk/2$), *CarrierFreq* shall be programmed with the two's complement signed value $12 \cdot 2^{28} \cdot f_{ca} / f_c$, where f_c is the *CoreClk* frequency. The *CarrierFreq* register is double-buffered; a newly programmed value becomes effective at the start of the next Integration Epoch.

Table 12. *CarrierFreq* register

Bit	Field	Description
31- 0	CarrierFreq	Carrier frequency parameter

After reset the *CarrierFreq* register contains all zeroes.

CarrierShift (R/W)

The CarrierShift register shown in Table 13 allows to adjust the carrier phase, see page 21. To offset the carrier phase by $\cdot$ ($-180° < \alpha \, 180°$), CarrierShift shall be programmed by the two's complement signed value $3072. \cdot \alpha /360$, with the valid range being from -1536 to 1536. The CarrierShift register is double-buffered; a newly programmed value is added once to the 12 MSBs of the Carrier NCO phase register at the start of the next Integration Epoch.

Table 13. *CarrierShift* register

Bit	Field	Description
31 - 12	Reserved	
11 - 0	CarrierShift	Carrier phase offset increment

After reset the *CarrierShift* register contains all zeroes.

CodeFreq (R/W)

The CodeFreq register shown in Table 14 determines the chip rate and hence the code frequency, see page 21. To obtain the frequency fco (0 < fco < CoreClk/2), CodeFreq shall be programmed with the unsigned value 232 fco/FCC, where fc is the CoreClk frequency. The CodeFreq register is double-buffered; a newly programmed value becomes effective at the start of the next Integration Epoch.

Table 14. CodeFreq register

Bit	Field	Description
31 - 0	CodeFreq	Code frequency parameter

After reset the *CodeFreq* register contains the hexadecimal value 8000’0000, corresponding to a chip rate of half the *CoreClk* frequency.

Note: Setting the CodeFreq register to a value lower than hexadecimal 0200’0000 (corresponding to a chip rate of 1/64th of the CoreClk frequency) can cause undefined behaviour in combination with negative code shifts. For the same reason, it should not be set to a value higher than hexadecimal 7E00’0000 (corresponding to a chip rate of (1/2-1/ 64). CoreClk) in combination with normal positive code shifts, or 7000’0000 (corresponding to a chip rate of (1/2-1/8). CoreClk) in combination with large positive code shifts.

Warning: Programming the *CodeFreq* register to zero means that it will not be possible to reprogram the register, since an Integration Epoch will not occur and thus a new value will never become effective.

CodeShiftCtl (R/W)

The *CodeShiftCtl* register shown in Table 15 allows to adjust the code phase, see page 55. To shift the code phase the *ShiftSteps* field shall be programmed by the unsigned number of shift steps, and the *ShiftDir* field shall indicate the direction of the phase shift. The *ShiftMode* field determines whether the phase shift shall be applied immediately, at the beginning of the next Integration Epoch or repeatedly at the beginning of all subsequent Integration Epochs. The *StepSize* field selects a shift step size of either 1/64th or 1/8th of a chip, where the latter is only valid for positive code shifts.

Table 15. CodeShiftCtl register

Bit	Field	Description
31 - 14	Reserved	
13	StepSize	Size of the code phase shift step 0 = 1/64th chip (default) 1 = 1/8th chip (only valid for positive code phase shifts)
12- 11	ShiftMode	Code phase shift mode 00 = apply once, at the beginning of next Integration Epoch (default) 01 = apply repeatedly at the beginning of every Integration Epoch 10 = apply code phase shift immediately 11 = reserved
10	ShiftDir	Direction of code phase shift 0 = positive code phase shift (default) 1 = negative code phase shift
9 - 0	ShiftSteps	Number of code phase steps (0 1023, default 0)

After reset the *CodeShiftCtl* register contains all zeroes.

Note: Only one phase shift command can be in progress at any given time; no other code phase shifts are accepted while one code phase shift operation is ongoing, which can take up to 1024 *CoreClk* cycles. The AGGA-2 does not indicate whether a code phase shift operation is ongoing.

Note: For proper operation the Code NCO code-rate phase increment plus the effective code shift should never be less than zero or higher than *CoreClk*/2.

IntEpochLen (R/W)

The *IntEpochLen* register shown in Table 16 programs the Integration Epoch length and dead-time, see page 25. The *EpochLen* field determines the length of the Integration Epoch, expressed as an integer fraction or multiple of a C/A-code epoch (nominally 1 ms). The *EpochDeadTime* field determines the start of the dead-time window, specified as a number of C/A-code epochs before the end of the Integration Epoch, with a valid range from 0 (no dead-time) up to the Integration Epoch Length as programmed by the *EpochLen* field minus 2, with a maximum value of 15. A newly written value becomes effective at the start of the next Integration Epoch coinciding with a full C/A-code Epoch.

Table 16. IntEpochLen register

Bit	Field	Description
31 - 7	Reserved	
6 - 3	EpochDeadTime	Start of dead-time window, (0 15, default 0)
2 - 0	EpochLen	Integration Epoch length 000 = 1/4 C/A-code epoch 001 = 1/2 C/A-code epoch 010 = 1 C/A-code epoch (default) 011 = 2 C/A-code epochs 100 = 4 C/A-code epochs 101 = 5 C/A-code epochs 110 = 10 C/A-code epochs 111 = 20 C/A-code epochs

After reset the *IntEpochLen* register contains the binary value "0000010", corresponding to an Integration Epoch of one C/A-code epoch and no dead-time.

CodeSetting (R/W)

The *CodeSetting* register shown in Table 17 selects which GNSS C/A-code shall be generated, see page 25. GPS-type C/A-codes include the standard Space Vehicle (SV) codes as defined in RD1, and C/A-codes which have been allocated to Regional Augmentation (RA) systems such as EGNOS, WAAS and MSAS. If *GPSMode* = 1 the *InitG2* field determines the desired C/A-code. When *GPSMode* = 0 the *InitG2* field has no effect. A write to the *CodeSetting* register becomes effective immediately.

Table 17. CodeSetting Register

Bit	Field	Description			
31 - 11	Reserved	Select GPS or GLONASS C/A-code 0 = generate GLONASS C/A-code (default) 1 = generate GPS and other C/A-code			
10	GPSMode				
9 - 0	InitG2	Initial state for G2 shift register of GPS C/A-code generator (octal)			
		0337 = SV1 0644 = SV5 0151 = SV9 0013 = SV13 0621 = SV17 0031 = SV21 0034 = SV25 0650 = SV29 0032 = SV33 0064 = SV37	0157 = SV2 0322 = SV6 0273 = SV10 0005 = SV14 0310 = SV18 0014 = SV22 0016 = SV26 0324 = SV30 0064 = SV34	0067 = SV3 0646 = SV7 0135 = SV11 0002 = SV15 0144 = SV19 0714 = SV23 0007 = SV27 0152 = SV31 0643 = SV35	0033 = SV4 0323 = SV8 0027 = SV12 0001 = SV16 0062 = SV20 0071 = SV24 0003 = SV28 0065 = SV32 0321 = SV36
		1106 = RA115 1617 = RA119 1532 = RA123 0520 = RA127 0740 = RA131	1241 = RA116 1076 = RA120 1250 = RA124 1731 = RA128 1007 = RA132	0267 = RA117 1764 = RA121 0341 = RA125 0706 = RA129 0450 = RA133	0232 = RA118 0717 = RA122 0551 = RA126 1216 = RA130

Note: Programming the *CodeSetting* register resets the C/A-code Generator to its initial state, but does not affect the state of the Integration Timer. The first observables and correlation values after programming this register will therefore be invalid.

CorrMode (R/W)

The *CorrMode* register shown in Table 18 selects the code spacing and controls the operation of the Correlator Unit, see page 27 and page 28. The *CodeInSpace* and *CodeOutSpace* fields controls the spacing for channel slaving. The code phase spacings for despreading can be selected by the *ChipSpacingEP* and *ChipSpacingPL* fields. The *SpaceSel* bit specifies whether the spacing is in multiples of half chips or multiples of *CoreClk* cycles. The *SwitchPeriod* field programs the Spacing Sequencer, and when active the *SwitchId1* to *SwitchId3* fields are used to program the corresponding spacings. The read-only *SwitchId* field indicates which spacing setting was used during the last Integration Epoch. The *DespreadMode* field determines which code phase combinations are used to despread the base-band signal. A write to the *CorrMode* register becomes effective immediately.

Table 18. CorrMode register

Bit	Field	Description
31 - 25	Reserved	Despreader module mode 000 = despread with (Off, Off, Off) (default) 001 = despread with (P, E, L) 010 = despread with (P, E-L, Off) 011 = despread with (P, E-L, E) 100 = despread with (P, E-L, L) 101 = despread with (P, P, Off) in Hybrid Parallel-Multiplex mode 110 = despread with (P, Off, Off) 111 = despread with (B, Off, Off)
24 - 22	DespreadMode	Chip spacing selection E-P used during last Integration Epoch, as for SwitchId0 (read only)
21 - 20	SwitchId	Time multiplexed multipath sequence length 00 = no sequencing, continuously select *ChipSpacingEP* (default) 01 = sequence of 2 Integration Epochs 10 = sequence of 3 Integration Epochs 11 = sequence of 4 Integration Epochs
19 - 18	SwitchPeriod	Switch identifier 3, as for SwitchId0
17 - 16	SwitchId3	Switch identifier 2, as for SwitchId0
15 - 14	SwitchId2	Switch identifier 1, as for SwitchId0
13 - 12	SwitchId1	Chip spacing selection E - P (Switch identifier 0) 00 = select one space (tap 5) (default) 01 = select two spaces (tap 4) 10 = select four spaces (tap 2) 11 = select six spaces (tap 0)
11 - 10	ChipSpacingEP *(SwitchId*0)	Chip spacing selection P - L 00 = select one space (tap 7) (default) 01 = select two spaces (tap 8) 10 = select four spaces (tap 10) 11 = select six spaces (tap 12)
9 - 8	ChipSpacingPL	
7	Reserved	
6 - 4	CodeOutSpace	Code Delay Line output tap selection 000 = zero space from *P* (tap 6) (default) 001 = one space from *P* (tap 7) 010 = two spaces from *P* (tap 8) 011 = three spaces from *P* (tap 9) 100 = four spaces from *P* (tap 10) 101 = five spaces from *P* (tap 11) 110 = six spaces from *P* (tap 12) 111 = no delay, connect CodeIn directly to CodeOut

3 - 1	CodeInSpace	Code Delay Line input tap selection. **Note:** see page 73 for a known implementation error. 000 = seven spaces to *P* (tap 0) (default) 001 = six spaces to *P* (tap 1) 010 = five spaces to *P* (tap 2) 011 = four spaces to *P* (tap 3) 100 = three spaces to *P* (tap 4) 101 = two spaces to *P* (tap 5) 110, 111 = Reserved
0	SpaceSel	Code Delay Line clock selection 0 = spacing in number of *CoreClk* cycles (default) 1 = spacing in number of half code chips (see note below)

After reset the *CorrMode* register contains all zeroes.

Note: Due to a design error code spacing in half code chips does not work as intended, see page 74.

Note: For proper operation there should never be less than 1 space between *CodeIn* and E.

CarrierObs (R)

The *CarrierObs* register shown in Table 19 is updated with new instantaneous values of the Carrier Cycle Count *CaCycleCount* and the 12 MSBs of the Carrier Phase *CaPhase* at the end of each Measurement Epoch, see page 22. The valid range of *CaCycleCount* is from -2048 to 2047 carrier cycles, while for *CaPhase* it is from -1536 to 1536, corresponding to a carrier phase fraction of *CaPhase* · 360°/3072.

Table 19. CarrierObs register

Bit	Field	Description
31 - 28	Reserved	
27 - 16	CaCycleCount	CaCycleCount
15 - 12	Reserved	
11 - 0	CaPhase	Carrier cycle fraction

After reset the *CarrierObs* register has an undefined value.

Note: Since the Carrier Cycle Count only represents the LSBs of the actual number of carrier cycles, in some cases, such as for establishing Doppler compensation, the firmware must calculate the total number of carrier cycles.

CodeObs (R)

The *CodeObs* register shown in Table 20 is updated with new instantaneous values of the C/A-code Epoch Count *CodeEpochCn*t, the C/A-code Chip Count *ChipCount* and the 16 MSBs of the Code Phase *CodePhase* at the end of each Measurement Epoch, see page 24 and page 26. The code phase fraction is calculated as *CodePhase* · 360°/65536. The value of the *CodeEpochCnt* field is the number of C/A-code epochs specified by the *IntEpochLen* register minus one minus the number of C/A-code epochs that have occurred since the start of the last Integration Epoch (i.e., it counts upwards starting from zero), with the range being from zero to the number of C/A-code epochs specified by *IntEpochLen* minus one. The 10-bit C/A-code Chip Count represents the number of C/A-code chips since the start of the current C/A-code epoch, with a range from 0 to 1023 for GPS and 0 to 511 for GLONASS.

Table 20. CodeObs register

Bit	Field	Description
31	Reserved	
30 - 26	CodeEpochCnt	C/A-code Epoch Count since last Integration Epoch
25 - 16	ChipCount	C/A-code Chip Count since last C/A-code Epoch
15 - 0	CodePhase	Code Phase fraction

After reset the *CodeObs* register has an undefined value.

Note: The total number of C/A-code epochs should be counted by the firmware to form the complete one second pseudorange.

Note: When the Integration Timer is programmed for 1/4, 1/2 or 1 C/A-code epoch, as can be useful during acquisition, the Code Epoch Count is invalid.

Correlation Value Registers (R)

The *CorrValue* fields of the *CorrValue0I* register shown in Table 21 and the

*CorrValue0*Q, *CorrValue1*I, *CorrValue1*Q, *CorrValue2I* and *CorrValue2Q* shown in Table 22 are updated with the correlation results at the end of each Integration Epoch, see page 32. Each correlation value is a 22-bit two's complement signed value, with the sign in bit 31. The *OverRun* bit is set when the *CodeFreq* register was not written before a new Integration Epoch occurred.

Table 21. CorrValue0I register

Bit	Field	Description
31 - 10	CorrValue	Correlation value
9 -1	Reserved	
0	OverRun	Indicates whether previous values were not read out in time

Table 22. CorrValue0Q, CorrValue1- and CorrValue2- registers

Bit	Field	Description
31 - 10	CorrValue	Correlation value
9 -1	Reserved	

After reset the *CorrValue* registers have an undefined value.

Note: For "E-L" despreading the resulting correlation values will be divided by two. For correlators that are not enabled the corresponding correlation values will be undefined.

P-code Unit Registers

PCodeSetting (R/W)

The *PCodeSetting* register shown in Table 23 controls the operation of the P-code Generator, see page 34. Programming the *SVSel* field to zero selects GLONASS operation, while programming it with a GPS Space Vehicle number selects GPS operation, with the valid range being 1 to 37. When GPS operation has been selected the *ZCount* field determines to which X1 epoch the P-code Generator is initialised, with a valid range from 0 to 403'199. The first P-code chip generated after completed initialisation and handover will be the first chip of the X1 epoch corresponding to a Z-count of the programmed value + 1. Programming the maximum value 403199 will lead to the P-code

generator starting at Z-count 0. A write to the *PCodeSetting* register becomes effective immediately.

Table 23. PCodeSetting register

Bit	Field	Description
31 -25	Reserved	
9 -1	SVSel	SV selection for GPS P-codes 0 = generate GLONASS P-code (default) 1 to 37 = generate GPS P-code for Space Vehicle number *SVSel*
18 -0	ZCount	Desired Z-count - 1 (default value 0)

After reset the *PCodeSetting* register contains all zeroes.

Note: Writing to the *PCodeSetting* register interrupts the P-code Generator. Programming a GPS P-code setting triggers the P-code Generator initialisation which takes 15'345'000 P-code chips to complete. Once started the initialisation can not be interrupted.

HandOver (W)

The *HandOver* register controls the start of the P-code Generator after it has been initialised, see page 37. The P-code Generator is started at the start of the first Integration Epoch after a write to the *HandOver* register.

Table 24. *HandOver* register

Bit	Field	Description
31 -0	Reserved	

Note: If handover is performed before the initialisation for GPS has been started or has been completed, the handover might lead to the P-code Generator being started at some Integration Epoch with an undefined code being generated.

CycleDiff (R/W)

The *CycleDiff* register shown in Table 25 programs the initial delay between the two P-code sequences generated by the P-code Unit, see page 37. The *L1LagsL2* field specifies whether the P-code for L1 or for L2 shall be delayed. The *CycleDiffCnt* field determines the delay expressed as an integer number of P-code chips, with a valid range from zero to ten when L2 is lagging L1, and zero to two when L1 is lagging L2. The *CycleDiff* register is double-buffered; after a value has been programmed it becomes effective once at the start of the next Integration Epoch.

Table 25. *CycleDiff* register

Bit	Field	Description
31 -5	Reserved	
4	L1LagsL2	Indicates whether the L1 or L2 P-code is delayed 0 = L2 is lagging L1 (default) 1 = L1 is lagging L2
3 - 0	CycleDiffCnt	Cycle difference count Initial delay expressed in P-code chip periods

After reset the *CycleDiff* register contains all zeroes. However, the delay between L1 and L2 initially depends on the order the corresponding CaP-channels were enabled.

Note: The behaviour is not defined for an out-of-range value of the *CycleDiffCnt* field.

PCodeObs (R)

The *PCodeObs* register shown in Table 26 is updated with new instantaneous values of the L1-L2 Cycle Difference value *CycleDiffObs* and the P-code Chip Count *PChipCount* at the end of each Measurement Epoch, see page 38. The *CycleDiffObs* field contains the integer part of the accumulated phase difference between the L1 and L2 P-code sequences. It contains a 5-bit value in sign/magnitude format with a valid range from -2 to +10, where a positive value (the MSB is zero) indicates that the L2 signal lags the L1 one. The 24-bit P-code Chip Count represents the number of L1 P-code chips since itwas last reset. The counter is reset at P-code Handover and when it has reached a count equal to the number of P-code chips in one second (5'110'000 P-code chips for GLONASS, 10'230'000 P-code chips for GPS).

Table 26. PCodeObs register

Bit	Field	Description
31 -5	Reserved	
4	L1LagsL2	Indicates whether the L1 or L2 P-code is delayed 0 = L2 is lagging L1 (default) 1 = L1 is lagging L2

After reset the *PCodeObs* register has an undefined value.

WEdgeCtl (R/W)

The *WEdgeCtl* register shown in Table 27 configures the W-edge Generator, see page 38. The *A* and *B* parameters each have a valid range from 11 to 31, where the number of P-code chips for one W-code chip is the same as the programmed value. The *M* and *N* parameters can each have a value between 0 and 63 W-code chips. The W-edge Generator is disabled when *M* is zero. The *S* parameter can have a value between 0 and 31 P-code chips. A write to the *WEdgeCtl* register becomes effective immediately.

Table 27. *WEdgeCtl* register

Bit	Field	Description
31 -27	Reserved	
26 - 22	S	Number of P-code chips before starting the W-edge generator
21 - 16	N	Number of W-code chips using the *B* length parameter
15 - 10	M	Number of W-code chips using the *A* length parameter
9 - 5	B	Second number of P-code chips for one W-code chip
4 - 0	A	First number of P-code chips for one W-code chip

After reset the *WEdgeCtl* register contains all zeroes, disabling the W-edge Generator.

Note: Enabling the W-edge Generator configures the corresponding dual-frequency channel for semi-codeless operation.

Threshold (R/W)

The *Threshold* register shown in Table 28 determines the absolute threshold levels for detecting a W-code chip for the two CaP-channels, with a range from 0 to 255, see page 39. A write to the *Threshold* register becomes effective immediately.

Table 28. *Threshold* register

Bit	Field	Description
31 - 16	Reserved	
15 - 18	Threshold2	Threshold level for W-estimates to the L2 CaP-channel
7 0	Threshold1	Threshold level for W-estimates to the L1 CaP-channel
15 - 10	M	Number of W-code chips using the *A* length parameter

After reset the *Threshold* register contains all zeroes.

Decisive (R)

The *Decisive* register shown in Table 29 indicates the number of decisive W-code chips that was detected during the last Integration Epoch for the two CaP-channels of a dual-frequency channel, see page 39. Each of the fields has a valid range from 0 to 16383.

Table 29. Decisive register

Bit	Field	Description
31 - 30	Reserved	
29 16	Decisive2	Number of decisive W-code chips for the L2 CaP-channel
15 14	Reserved	
13 - 0	Decisive1	Number of decisive W-code chips for the L1 CaP-channel

After reset the *Decisive* register has an undefined value.

Note: The value of the *Decisive1* field is generated by the W-chip Estimator for the L1 CaP-channel, and it is related to the primary correlation values originating from the L2 CaP-channel, and vice versa.

Time Base Generator Registers

TimeBaseSetting (R/W)

The *TimeBaseSetting* register shown in Table 30 controls the operation of the Time Base Generator, see page 42. The *PPSDivRatio* field determines the division ratio for the *PPS* output, and has a range from 0 to 16383. To obtain the frequency f_{PPS}, *PPSDivRatio* shall be programmed with the integer value f_e/f_{PPS} -1, where f_e is the *EpochClk* frequency. The *PPSEnable* and *MEOEnable* fields control whether the *PPS* and *MEO* outputs are enabled. The *MEODivRatio* field determines the division ratio for the *MEO* output, and has a range from 0 to 63. To obtain the frequency f_{ME}, *MEODivRatio* shall be programmed with the integer value f_e/f_{ME} /20-1. A new value written to the *MEODivRatio* field becomes effective at the start of the next *MEO* cycle. A new value written to the *PPSDivRatio* field becomes effective at the start of the next *PPS* cycle. The enable bits become effective immediately.

Table 30. TimeBaseSetting register

Bit	Field	Description
31 - 30	Reserved	
29 - 16	PPSDivRatio	Division ratio of *PPS* divider - 1
15	PPSEnable	Enable *PPS* output 0 = disable *PPS* (default) 1 = enable *PPS*
14	MEOEnable	Enable *MEO* output 0 = disable *MEO* (default) 1 = enable *MEO*
13 - 6	Reserved	
5 - 0	MEODivRatio	Division ratio of the Measurement Epoch divider - 1

After reset the *TimeBaseSetting* register contains all zeroes, disabling the *PPS* and *MEO* outputs. This register is not reset by a write access to the ChipReset register.

EpochClkDiv (R/W)

The *EpochClkDiv* register shown in Table 31 controls the *EpochClk* frequency, see page 42. To obtain the frequency f e, *EpochClkDiv* shall be programmed with the value f c /f e -1, where f c is the *CoreClk* frequency, expressed as a 26-bit fixed-point value having a 16-bit integer part and a 10-bit fractional part. The valid range of *EpochClkDiv* is from 5 to 65534. A newly programmed value becomes effective at the next EpochClk clock cycle.

Table 31. EpochClkDiv register

Bit	Field	Description
31 - 26	Reserved	
25 - 10	*EpochClkDiv*	Integer part of *(EpochClk* division ratio) -1
9 -0		Fractional part of *(EpochClk* division ratio) -1

After reset this register contains hexadecimal 200'0000, corresponding to an *EpochClk* frequency of *CoreClk*/32769. This register is not reset by a write to the *ChipReset* register.

Note: For non-integer division ratios the number of *CoreClk* cycles per *EpochClk* can differ by one *CoreClk* cycle.

Note: When the *EpochClk* division ratio is adjusted this will affect the length of the Measurement Epoch, which might need to be compensated for by the firmware.

PPSMEState (R)

The *PPSMEState* register shown in Table 32 is updated with the state of the *PPS* divider *PPSDivState* and the Measurement Epoch Period *MEPeriod* at the end of every Measurement Epoch, see page 42. The *PPSDivState* field contains the number of remaining *EpochClk* cycles until the *PPS* output will be asserted, where the value zero means at the start of the next *EpochClk* cycle. This value can be used by the microprocessor to determine the remaining number of *EpochClk* cycles until the *PPS* strobe will be asserted. The *MEPeriod* field contains the 16 LSBs of the number of *CoreClk* cycles since the previous Measurement Epoch, which can be used for calculating the exact number of *CoreClk* cycles for the last Measurement Epoch.

Table 32. PPSMEState register

Bit	Field	Description
31 - 26	Reserved	
29 - 16	PPSDivState	PPS divider state
15 - 0	MEPeriod	LSBs of *CoreClk* cycle count since last Measurement Epoch

After reset the *PPSMEState* register has an undefined value. This register is not reset by a write access to the *ChipReset* register.

Antenna Switch Controller Registers

AntSwitchMode (R/W)

The *AntSwitchMode* register shown in Table 33 controls the operation of the Antenna Switch Controller, see page 44. The read-only *AntSwitchId* field indicates which *ASC* output was asserted during the last Antenna Switch Epoch, with the range from 0 to 3. The *ASEODivRatio* field determines the division ratio for the *ASEO* output, and has a range from 0 to 31. To obtain the frequency f ASE, *ASEODivRatio* shall be programmed with the integer value f e /f ASE -1, where f e is the *EpochClk* frequency. The *ASEOEnable* field controls whether the *ASEO* output is enabled. The *ASCEnable0* to *ASCEnable3* bits enables which *ASC* outputs are included in the switching sequence. A write to the *AntSwitchMode* register becomes effective immediately.

Table 33. AntSwitchMode register

Bit	Field	Description
31 - 26	Reserved	
17 16	AntSwitchId	ASC output asserted during last Antenna Switch Epoch (read only)
15	ASEOEnable	Enable *ASEO* output 0 = disable *ASEO* (default) 1 = enable *ASEO*
14 12	Reserved	
11 7	ASEODivRatio	(Antenna Switch Epoch division ratio) -1
6 4	Reserved	
3	ASCEnable3	Antenna Switch Controller enable 0 = disable *ASC3* (default) 1 = include *ASC3* in the switching sequence
2	ASCEnable2	Idem for *ASC2*
1	ASCEnable1	Idem for *ASC1*
0	ASCEnable0	Idem for *ASC0*

After reset *AntSwitchMode* contains all zeros.

System Support Function Registers

ChipReset (W)

Writing to the *ChipReset* register shown in Table 34 resets the AGGA-2, with exception of the Epoch Clock Divider, the MEO Divider, the PPS Divider and the GenClk divider, see page 47.

Table 34. ChipReset register

Bit	Field	Description
31 - 0	Reserved	

InP (R)

The *InP* register shown in Table 35 contains the instantaneous value of the *InP19 0* input pins, see page 47**.**

Table 35. InP register

Bit	Field	Description
31 - 20	Reserved	
19 - 0	InP	Instantaneous value of the *InP* inputs

After reset the *InP* register contains the values of the *InP* inputs.

Note: Since intended for slowly varying signals, the signals for the Input Port are not latched when accessed, and the value read from the *InP* register can change during the access. Be reading the *InP* register until the two last values read are identical a coherent value for the entire register can be obtained.

OutP (R/W)

The *OutP* register shown in Table 35 sets the value of the *OutP* outputs, see page 47.

Table 36. OutP register

Bit	Field	Description
31 - 12	Reserved	
11 - 0	OutP	Value of the *OutP* inputs

After reset the *OutP* register contains all zeroes.

CodeOutput (R/W)

The *CodeOutput* register shown in Table 37 controls the output of a GNSS code sequence from any channel in the Channel matrix, which can be used for GNSS transmission, calibration or debugging, see page 47 The *CodeDelay* field specifies the number of *CoreClk* cycles the code sequence will be delayed w.r.t. the *CodeOut* signal, with a range from 3 to 18. *CodeDelay* shall be programmed with the desired delay minus three. The *CodeEnable* bit determines whether the selected code sequence is output on two *OutP* outputs. The *Channel* field specifies from which channel the code sequence shall be taken, with a valid range from zero to eleven.

Table 37. CodeOutput register

Bit	Field	Description
31 - 9	Reserved	
8 - 5	CodeDelay	(Code sequence delay from *CodeOut*) - 3
4	CodeEnable	Enables the output of a GNSS code sequence and Integration Epoch 1 = enable code sequence output 0 = disable code sequence output
3 - 0	Channel	Selection of which channel to output the GNSS signal

After reset the *CodeOutput* register contains all zeros.

Note: For proper operation *CodeDelay* shall correspond to less than one code chip.

Interrupt Controller Registers

IntMask (R/W)

The *IntMask* register shown in Table 38 allows separate masking of each interrupt that can be generated by the AGGA-2, see page 46 Each bit in the *IntMask* register corresponds to a bit in the same position in the interrupt status register *IntStatu*s. A value of zero will assert the *IntOut* output when the corresponding interrupt is generated, while a value of one will mask it.

Table 38. IntMask register

Bit	Field	Description
31 - 25	Reserved	
24 - 13	IntEpochMask	Integration Epoch interrupt masks for channel 11 to 0, where bit 24 corresponds to channel 11 and bit 13 to channel 0
12	AntEpochMask	Antenna Switch Epoch interrupt mask
11	LevelMask	Signal Level Detector interrupt mask
10	MeasEpochMask	Measurement Epoch interrupt mask
9	PPSMask	Pulse-Per-Second interrupt mask
8 - 2	Reserved	
10	IntInMask	Daisy-chain interrupt mask 1 0

After reset the *IntMask* register bits 9 to 24 are all one, masking all internal interrupt sources, while bits 0 and 1 are zero to enable the external daisy-chain interrupts.

IntStatus (R/W)

The *IntStatus* register shown in Table 38 provides the last acknowledge status of all internal interrupts in the AGGA-2, together with the instantaneous status of the external daisy-chain interrupt inputs, see page 46. The *IntStatus* register includes the status for all interrupts, regardless of whether masked or not. Writing this register (with arbitrary data) will transfer the status of all internal interrupts to the *IntStatus* register within eight *CoreClk* cycles, and reset the internal shadow register. A value of one in a bit of the *IntStatus* register indicates that the corresponding interrupt has been generated.

Table 39. IntStatus register

Bit	Field	Description
31 - 25	Reserved	
24 - 13	IntEpochInt	Integration Epoch interrupt status for channel 11 to 0, where bit 24 corresponds to channel 11 and bit 13 to channel 0
12	AntEpochInt	Antenna Switch Epoch interrupt mask
11	*LevelInt*	Signal Level Detector interrupt status
10	MeasEpochInt	Measurement Epoch interrupt status
9	PPSInt	Pulse-Per-Second interrupt status
8 - 2	Reserved	
10	IntIn	Daisy-chain interrupt status 1 0

After reset the *IntStatus* register contains all zeros, except for the daisy-chain interrupts which reflects the values at the *IntIn0* and *IntIn1* inputs.

Power Reduction of Unused Functions

While many functions in the AGGA-2 are automatically powered-down when not used, there are some functions that need to be programmed into a state where the power consumption is minimum. The default values for all registers after a reset have been selected for minimizing the power consumption.

To reduce the power consumption when a particular unit or channel is not used, the following values should be programmed:

- The *CarrierFreq* register should be programmed to 0 when the channel is disabled or the carrier is slaved.
- The *CodeFreq* register should be programmed to hexadecimal 8000'0000 when the channel is disabled or CA-code is slaved.
- The *DespreadMode* field of the *CorrMode* register should be programmed to “Off, Off, Off” when the channel is disabled.

Programming Restrictions

The reserved fields of a register can be programmed with arbitrary values without any impact on the AGGA-2. When read, reserved fields and registers that are marked as write-only have an undefined value. Writing to registers that are marked as read-only should be avoided.

All registers and fields associated with a unit that is disabled due to channel slaving have undefined values when read, including the case of a P-code Unit when its signals are not used by the two associated CaP-channels. After a channel has been disabled and then is enabled again, the contents of all read-only registers are undefined. Read/write registers keep the value that was last written to them.

The value read back from registers that are double-buffered reflect the last programmed value, rather than the value that is currently effective.

After a write access to the AGGA-2 no new write access should be made for at least four *CoreClk* cycles to allow the value to allow sufficient time to properly synchronize the write access on the chip. Reading back a read/write register after a new value has been written to it will provide the new value within five *CoreClk* cycles. After a channel has been enabled by writing to the *ChannelMode* register, no other register of that channel should be written for five *CoreClk* cycles. After an interrupt acknowledge by writing to

the *IntStatus* register it can take up to eight *CoreClk* cycles before the *IntStatus* register contains the updated status.

Writing to a double-buffered register while its value is becoming effective (a few *CoreClk* cycles before the associated Integration Epoch occurs, as indicated by the corresponding interrupt) can cause undefined behaviour. Reading a double-buffered register while its values are being updated (a few *CoreClk* cycles before the Measurement Epoch or the associated Integration Epoch occurs, as indicated by the corresponding interrupt) can yield invalid values.

INTERFACE AND SIGNAL DESCRIPTION

The AGGA-2 has 62 input signals, 21 output signals and 32 bi-directional signals. Five pins are not connected. Together with 40 power supply pins this gives a total pin count of 160. Active low signals are indicated by being overlined.

Front-end Interface, 16 pins

ADCIn7[1:0] to ADCIn0[1:0]: ADC Inputs (I)

These TTL inputs form the eight *ADCIn* inputs, each providing 2-bit samples to the AGGA-2 in the formats defined in Table 1. When the input signal is in real format these inputs are sampled both on the rising and falling edges of *CoreCl*k. When the input signal is in complex format these inputs are sampled on the rising edge of *CoreCl*k.

Time Base Generator Interface, 3 pins

MEI: Measurement Epoch Input (I)

After synchronisation and detection of the rising edge this CMOS input generates the Measurement Epoch in the AGGA-2. As the input is internally synchronised to *CoreCl*k, it may be asynchronous, but for fully deterministic operation it should be stable on the rising *CoreClk* edge. The Synchroniser incurs a delay of three *CoreClk* cycles.

MEO: Measurement Epoch Output (O)

This CMOS output is asserted (logical one) to generate a new Measurement Epoch. It is asserted for 128 *CoreClk* periods. It is intended to be connected to the *MEI* input of one or more AGGA-2s. *MEO* changes state on the rising *CoreClk* edge.

PPS: Pulse-Per-Second Strobe (O)

This CMOS output is asserted (logical one) to indicate a reference time. It is asserted for 128 *CoreClk* periods. *PPS* changes state on the rising *CoreClk* edge.

Antenna Switch Controller Interface, 6 pins

ASEI: Antenna Switch Epoch Input (I)

After synchronisation and detection of the rising edge this CMOS input generates the Antenna Switching Epoch in the AGGA-2. As the input is internally synchronised to *CoreCl*k, it may be asynchronous, but for fully deterministic operation it should be stable on the rising *CoreClk* edge. The Synchroniser incurs a delay of three *CoreClk* cycles.

ASEO: Antenna Switch Epoch Output (O)

This CMOS output is asserted (logical one) to generate a new Antenna Switching Epoch. It is asserted for 128 *CoreClk* periods. It is intended to be connected to the *ASEI* input of one or more AGGA-2s. *ASEO* changes state on the rising *CoreClk* edge.

ASC3 to ASC0: Antenna Switch Control (O)

These CMOS outputs are asserted (logical one) to select one of up to four antennas. Only one *ASC* output is asserted at any time, and it remains asserted until the next one is asserted. *ASC* changes state on the rising *CoreClk* edge.

System Support Functions Interface, 39 pins

ClkIn: Separate Clock Input (I)

This separate TTL clock input is used for the generation of the *GenClk* clock output.

GClkDiv2 to GClkDiv0: General Purpose Clock Division Ratio (I)

These static CMOS inputs define the *ClkIn* division ratio for generating the General Purpose clock. *GClkDiv2* is the most significant bit. A value of zero disables the *GenClk* divider, while for any other binary value in the range one to seven, *GenClk* is generated as *ClkIn* divided by (the value of *GClkDiv* + 1).

GenClk: General Purpose Clock (O)

This CMOS output is intended as a general purpose clock, with the frequency programmable by the *GClkDiv* inputs. It changes state on the rising *ClkIn* edge.

InP19 to InP0: Input Port (I)

The values of these asynchronous TTL inputs can be read by the microprocessor from the *InP* register. The inputs are latched on the rising *CoreClk* edge.

These inputs are shared with the Front-end Interface. When used as redundant inputs in complex sign/magnitude format these inputs are sampled on the rising edge of *CoreClk.*

OutP11 to OutP0: Output Port (O)

These CMOS outputs reflect the values of the corresponding bits in the *OutP* register. *OutP* changes state on the rising *CoreClk* edge.

CoreClk: AGGA-2 Internal Clock (I)

This TTL clock input provides the AGGA-2 internal clock.

Reset: AGGA-2 Reset (I)

When asserted (logical zero) this asynchronous Schmitt trigger input resets the AGGA-2.

Microprocessor Interface, 50 pins

CS: Chip Select (I)

This active low TTL asynchronous input controls the access to the AGGA-2. When asserted (logical zero) it enables the microprocessor interface of the AGGA-2 and indicates that a valid address is available.

RD: Read Control (I)

This asynchronous TTL input indicates a read operation when asserted (logical zero). The data from the AGGA-2 will be placed on the data bus when *RD* is asserted. When deasserted the data bus will be tristated.

WR: Write Control (I)

This asynchronous TTL input indicates a write operation when asserted (logical zero), which also tristates the data bus. The address and data from the microprocessor are latched on the rising edge of *W*R. After a write access to the AGGA-2 no new write access should be made for at least three *CoreClk* cycles, to allow the value to properly propagate to the internal destination.

ARange1 to ARange0: Additional Chip Select Setting (I)

These static CMOS inputs define for which value on the *A9* and *A8* address signal the AGGA-2 can be accessed. *ARange1* is the most significant bit.

A9 to A0: Address Bus (I)

These asynchronous TTL inputs provide the address for accessing the AGGA-2 registers. *A9* is the most significant bit. The address bus should be stable when *RD* or *WR* is asserted.

D31 to D0: Data Bus (I/O)

These bi-directional TTL signals provide the data to and from the AGGA-2. *D31* is the most significant bit.

IntIn1 to IntIn0: Interrupt Request Input (I)

Each of these asynchronous TTL inputs indicates an interrupt request from another device, to subsequently be provided to the microprocessor via the *IntOut* output.

IntOut: Interrupt Request Output (O)

This CMOS output indicates an interrupt issued by the AGGA-2. *IntOut* is asynchronous while *Reset* is asserted, and after reset it changes state on the rising *CoreClk* edge.

Test and Power, 41 pins

Test (I)

The test input is sets the AGGA-2 in production test mode. It should always be tied to logic zero.

VDDA, VSSA: Array Power and Ground

The AGGA-2 has nine pins for the Array Power Supply (VDDA) and nine pins for the Array Ground (VSSA), allowing different supplies for the array than used for the inputs and output buffers.

VDDB, VSSB: Buffer Power and Ground

The AGGA-2 has eleven pins for the Buffer Power Supply (VDDB) and eleven pins for the Buffer Ground (VDDB), allowing different supplies for the array than used for the inputs and output buffers.

Note: The buffer power supply must always be the same or higher than the array supply.

STATE AFTER RESET

This section specifies the AGGA-2 state during and after reset.

Output Values During and After Reset

During reset by asserting the *Reset* input the AGGA-2 outputs *ME*O, *PP*S, *ASE*O, *ASC, Out*P, and *GenClk* are deasserted (logical zero). The *D* data bus is tristated. The interrupt output signal *IntOut* is a combinational function of the *IntIn0* and *IntIn1* inputs, with no impact from any internal interrupts.

After *Reset* has been deasserted *GenClk* operates as programmed by *GClkDi*v.

Operational Mode After Reset

Reset by asserting *Reset* places all channels in their off mode, with each register having an initial value as specified in “PROGRAMMING” on page 48

A write access to the *ChipReset* register will have the same effect, except that the generation of *GenClk* and the circuitry associated with the generation *PPS* output, including the Measurement Epoch, will not be affected.

After a reset, the AGGA-2 registers are initialised which takes 32 *CoreClk* cycles, during which they should not be accessed.

IMPLEMENTATION ASPECTS

Implementation Loss

The implementation losses in the AGGA-2 and its interfaces are estimated to be:

- 0.55 dB for the quantisation to 2 bits for the input signal (occurs outside the AGGA-2);
- 0.3 dB for the Real-to-Complex Converter, only when real inputs used;
- 0.4 dB for the Carrier Generator and the Final Down-converter;
- 0.2 dB for the Code Generator and the Correlator Unit for a code rate of *CoreCl*k/3, and 0.05 dB for a code rate of *CoreCl*k/6.

Internal Delays

The signal delay from the *ADCIn* inputs to the despreader is five *CoreClk* cycles for complex input signals and ten *CoreClk* cycles for real input signals.

The code delay from the Code NCO to the Code Delay Line input is three *CoreClk* cycles for C/A-code. For the L1 P-code it is six chips plus four *CoreClk* cycles, with an uncertainty of one chip, due to that the P-code generator is initialised at the P-code chip-rate whereas the Integration Epoch (from the CA-channel) can arrive any *CoreClk* cycle.

Code Polarity

In the AGGA-2 implementation, a chip value of logical one represents +1, while a logical zero represents -1. In the chosen representation of the sign/magnitude format the sign bit is logical one for a positive signal, c.f. Table 1. Since in the despreader the sign bit is XOR-ed with the chip value, the result is that for a positive input signal in phase with the code the correlation values will be negative. This has no impact in an actual GNSS receiver, since the correct code polarity is determined as part of the acquisition process.

Test Aspects

The ASIC test strategy was to use functional testing since it requires lower complexity and gives better detection of manufacturing defects than scan testing. The chip controllability was improved by techniques such as writing all channels simultaneously, acceleration of long counters, etc. Visibility was increased by reading all channels simultaneously, continuous monitoring of the despreader output and P-code Generator logic, etc. A fault coverage of 99.1% (counting faults on outputs only) was reached.

During the AGGA-2 development JTAG was listed as a desirable feature to facilitate board-level interconnection testing. However, it was not possible to implement it in the AGGA-2 due to schedule limitations, and the fact that it was only requested by one potential AGGA-2 user.

Known Design Errors and Suggested Work-arounds

During the initial evaluation of the AGGA-2 a number of discrepancies between the Data Sheet and the actual implementation has been identified, as described below.

Microprocessor Interface Write Access Errors

Data writes to the AGGA-2 registers sometimes result in incorrect values for some of the bits written. The incorrect values are in fact the data from the preceding write access to the AGGA-2. It occurs for all registers except for the *ChannelMode* registers. Although the origin of the error is a missing flip-flop for the write strobe (except for *ChannelMod*e), the resulting symptoms are the same as for a metastability problem.

Suggested work-around: By writing the desired data value to an unimplemented register (e.g. address hexadecimal FF) the internal data register is loaded with this value, without having an effect on the AGGA-2 operation. Writing the same data again, now to the intended register, ensures that all bits are correct, since they are either the value from the first or the second write access, which are the same. It is recommended to disable interrupts from the AGGA-2 during this period to avoid that a new write access takes place between the two write accesses with same data. Another approach is to synchronise the clocks of the microprocessor and the AGGA-2, with a phase relation where no write errors occur.

Code Delay Line Incorrect Spacing

The code spacing is incorrect when the Code Delay Line operates with code spacing in half chips. For code frequencies up to and including *CoreCl*k/6 the actual *CodeOut-Space* is one space (half a chip) shorter than what is specified in the *CorrMode* register (Table 18), i.e. *CodeOut* is taken from taps 5 to 11 instead of taps 6 to 12. Furthermore, for code frequencies up to and including *CoreCl*k/4 the delay from code in to Punctual is one *CoreClk* cycle shorter than what is specified. For higher code frequencies, the code spacing behaviour is complex and not further described here.

Suggested work-around: Only use spacing in half chips, typically for C/A-code acquisition, for code frequencies up to those stated above. If necessary, compensate for the error by modifying the values that are programmed to the *CorrMode* register.

P-code Incorrect at Start Of Week

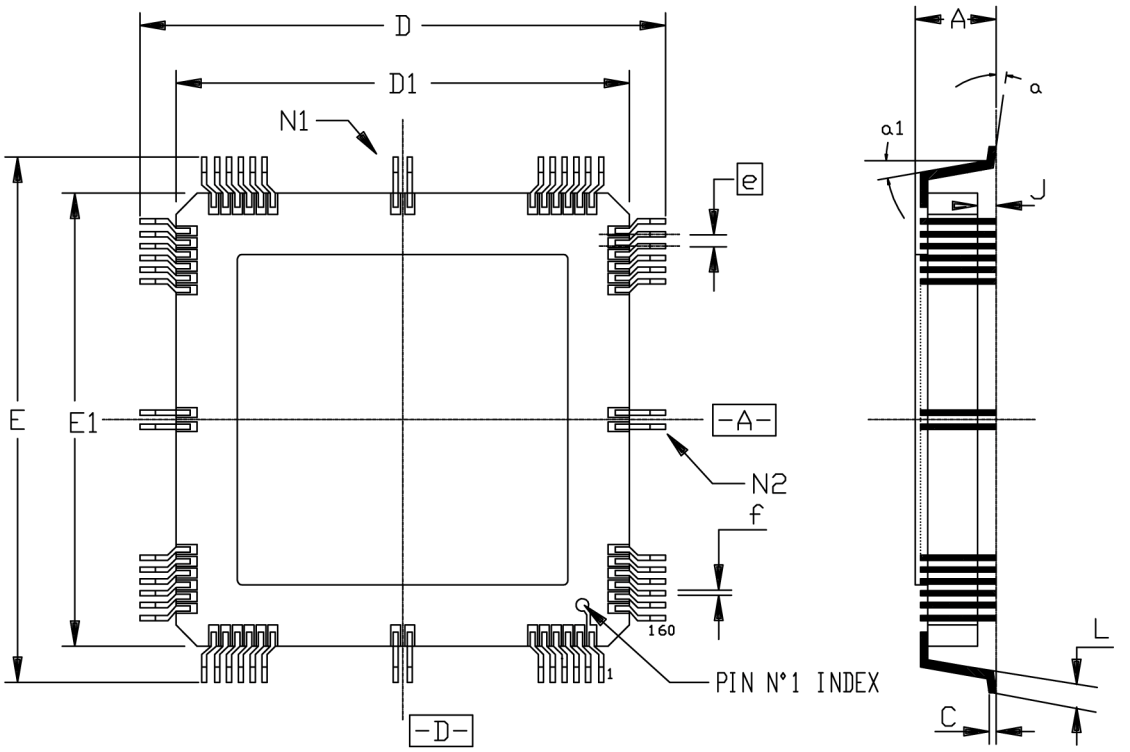
During dual-frequency operation, up to half of the GPS P-code chips generated during the 37 first chips at the start of the week are incorrect.

This error has insignificant impact on the operation of the AGGA-2 and can therefore be ignored.

Mechanical Specifications

The AGGA 2 will be packaged in an MQFPL160 package shown in Figure 24.

Figure 27. AGGA 2 mechanical data



	MM		INCH	
	Min	Max	Min	Max
A	2.44	3.60	.096	.142
C	0.15 TYP		.006 TYP	
D	31.93	32.67	1.257	1.286
D1	26.93	27.47	1.060	1.082
E	31.93	32.67	1.257	1.286
E1	26.93	27.47	1.060	1.082
e	0.65 BSC		.0256 BSC	
f	0.30 REF		.012 REF	
J	0.50	1.00	.020	.040
L	1.21	1.41	.047	.056
N1	40		40	
N2	40		40	
	a=2° ± 3°		a1=5° ± 3°	

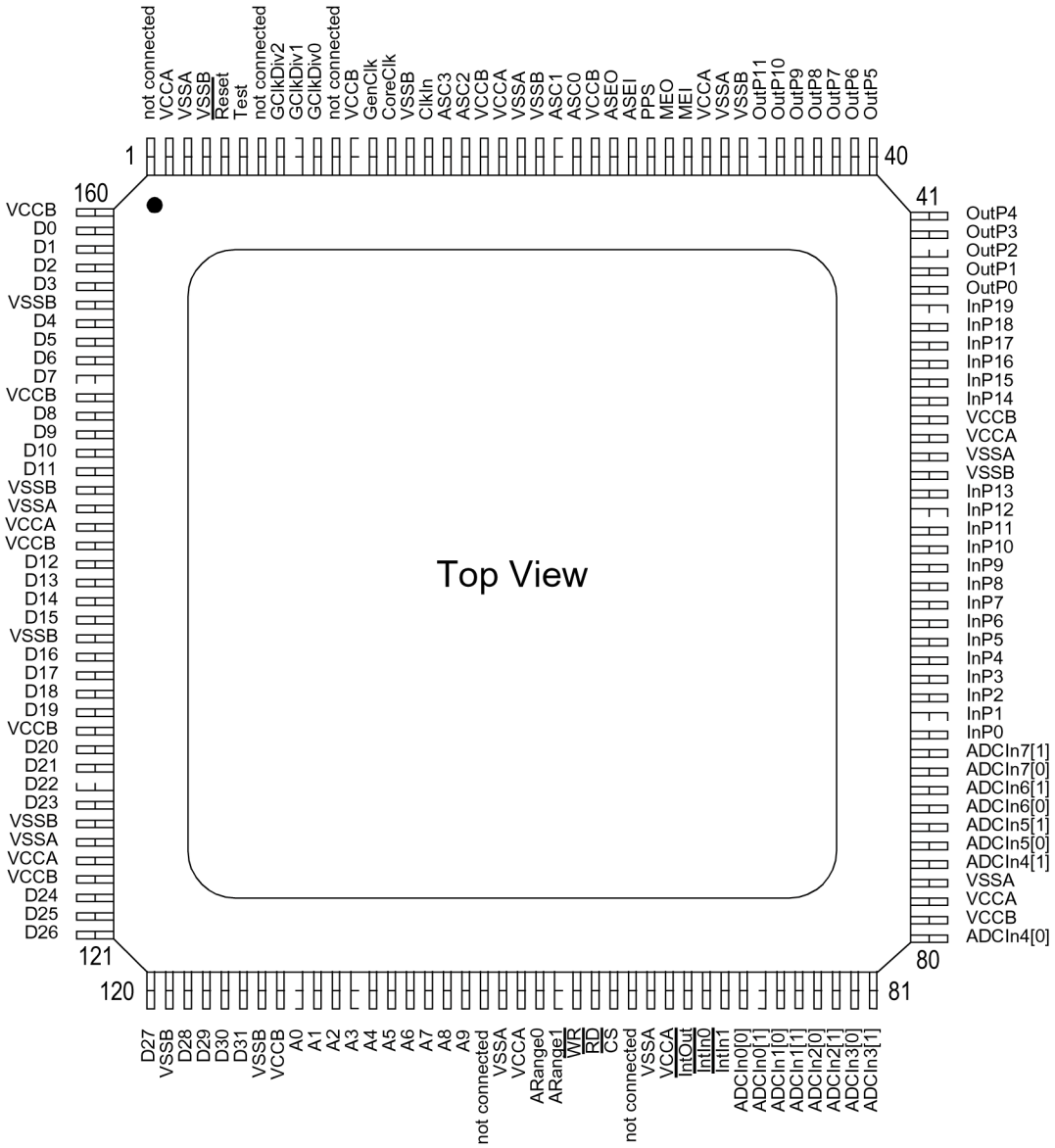
PACKAGE CODE : U13N U25N U41N U58N
INTERNAL CODE : FW

REV : F DATE : 19-05-98

Pin out

The AGGA 2 pin out is shown below.

Figure 28. AGGA 2 pinout



Pin list

The AGGA 2 pin list, giving the pad types and pad numbers, is shown in Table 43.

Table 40. AGGA 2 pin list

Name	Pad type	Pad number	Pin number	Pin type
-	-	-	1	Not connected
VCCA1	BVCCA	91	2	VDA
VSSA1	BVSSA	89	3	VSA
VSSB1	BVSSBENQ	86	4	VSB
ResetN	BTGCMOS	84	5	Input
Test	BINCMOS	82	6	Input
-	-	-	7	Not connected
GClkDiv_2_port	BINCMOS	77	8	Input
GClkDiv_1_port	BINCMOS	75	9	Input
GClkDiv_0_port	BINCMOS	72	10	Input
-	-	-	11	Not connected
VCCB1	BVCCB	68	12	VDB
GenClk	BOUT12	65	13	Output
CoreClk	BINTTL	63	14	Input
VSSB2	BVSSB	61	15	VSB
ClkIn	BINTTL	59	16	Input
ASC_3_port	BOUT12	56	17	Output
ASC_2_port	BOUT12	54	18	Output
VCCB2	BVCCBSNQ	52	19	VDB
VCCA2	BVCCA	49	20	VDA
VSSA2	BVSSA	47	21	VSA
VSSB3	BVSSBENQ	45	22	VSB
ASC_1_port	BOUT12	42	23	Output
ASC_0_port	BOUT12	40	24	Output
VCCB3	BVCCBP	38	25	VDB
ASEO	BOUT12	35	26	Output
ASEI	BINCMOS	33	27	Input
PPS	BOUT12	31	28	Output
MEO	BOUT12	28	29	Output
MEI	BINCMOS	26	30	Input
VCCA3	BVCCA	24	31	VDA

Name	Pad type	Pad number	Pin number	Pin type
VSSA3	BVSSA	21	32	VSA
VSSB4	BVSSBENQ	19	33	VSB
OutP_11_port	BOUT6	17	34	Output
OutP_10_port	BOUT6	14	35	Output
OutP_9_port	BOUT6	12	36	Output
OutP_8_port	BOUT6	10	37	Output
OutP_7_port	BOUT6	7	38	Output
OutP_6_port	BOUT6	5	39	Output
OutP_5_port	BOUT6	3	40	Output
OutP_4_port	BOUT6	382	41	Output
OutP_3_port	BOUT6	380	42	Output
OutP_2_port	BOUT6	378	43	Output
OutP_1_port	BOUT6	375	44	Output
OutP_0_port	BOUT6	373	45	Output
InP_19_port	BINTTL	371	46	Input
InP_18_port	BINTTL	368	47	Input
InP_17_port	BINTTL	366	48	Input
InP_16_port	BINTTL	364	49	Input
InP_15_port	BINTTL	361	50	Input
InP_14_port	BINTTL	359	51	Input
VCCB4	BVCCBSNQ	357	52	VDB
VCCA4	BVCCA	354	53	VDA
VSSA4	BVSSA	352	54	VSA
VSSB5	BVSSBENQ	350	55	VSB
InP_13_port	BINTTL	347	56	Input
InP_12_port	BINTTL	345	57	Input
InP_11_port	BINTTL	343	58	Input
InP_10_port	BINTTL	340	59	Input
InP_9_port	BINTTL	338	60	Input
InP_8_port	BINTTL	336	61	Input
InP_7_port	BINTTL	333	62	Input
InP_6_port	BINTTL	331	63	Input
InP_5_port	BINTTL	329	64	Input
InP_4_port	BINTTL	326	65	Input
InP_3_port	BINTTL	324	66	Input

Name	Pad type	Pad number	Pin number	Pin type
InP_2_port	BINTTL	322	67	Input
InP_1_port	BINTTL	320	68	Input
InP_0_port	BINTTL	317	69	Input
ADCIn7_1_port	BINTTL	315	70	Input
ADCIn7_0_port	BINTTL	313	71	Input
ADCIn6_1_port	BINTTL	310	72	Input
ADCIn6_0_port	BINTTL	308	73	Input
ADCIn5_1_port	BINTTL	306	74	Input
ADCIn5_0_port	BINTTL	303	75	Input
ADCIn4_1_port	BINTTL	301	76	Input
VSSA5	BVSSA	299	77	VSA
VCCA5	BVCCA	296	78	VDA
VCCB5	BVCCBP	298	79	VDB
ADCIn4_0_port	BINTTL	292	80	Input
ADCIn3_1_port	BINTTL	286	81	Input
ADCIn3_0_port	BINTTL	284	82	Input
ADCIn2_1_port	BINTTL	282	83	Input
ADCIn2_0_port	BINTTL	280	84	Input
ADCIn1_1_port	BINTTL	278	85	Input
ADCIn1_0_port	BINTTL	276	86	Input
ADCIn0_1_port	BINTTL	273	87	Input
ADCIn0_0_port	BINTTL	271	88	Input
IntInN_1_port	BINTTL	269	89	Input
IntInN_0_port	BINTTL	266	90	Input
IntOutN	BOUT12	264	91	Output
VCCA6	BVCCA	262	92	VDA
VSSA6	BVSSA	259	93	VSA
-	-	-	94	Not connected
CSN	BINTTL	255	95	Input
RDN	BINTTL	253	96	Input
WRN	BINTTL	250	97	Input
ARange_1_port	BINCMOS	248	98	Input
ARange_0_port	BINCMOS	246	99	Input
VCCA7	BVCCA	243	100	VDA
VSSA7	BVSSA	241	101	VSA

Name	Pad type	Pad number	Pin number	Pin type
-	-	-	102	Not connected
A_9_port	BINTTL	236	103	Input
A_8_port	BINTTL	234	104	Input
A_7_port	BINTTL	232	105	Input
A_6_port	BINTTL	229	106	Input
A_5_port	BINTTL	227	107	Input
A_4_port	BINTTL	225	108	Input
A_3_port	BINTTL	222	109	Input
A_2_port	BINTTL	220	110	Input
A_1_port	BINTTL	218	111	Input
A_0_port	BINTTL	215	112	Input
VCCB6	BVCCBSNQ	213	113	VDB
VSSB6	BVSSBENQ	211	114	VSB
D_31_port	BIOTQ	208	115	I/O
D_30_port	BIOTQ	206	116	I/O
D_29_port	BIOTQ	204	117	I/O
D_28_port	BIOTQ	201	118	I/O
VSSB7	BVSSBP	199	119	VSB
D_27_port	BIOTQ	197	120	I/O
D_26_port	BIOTQ	188	121	I/O
D_25_port	BIOTQ	186	122	I/O
D_24_port	BIOTQ	184	123	I/O
VCCB7	BVCCBSNQ	181	124	VDB
VCCA8	BVCCA	179	125	VDA
VSSA8	BVSSA	177	126	VSA
VSSB8	BVSSBENQ	174	127	VSB
D_23_port	BIOTQ	172	128	I/O
D_22_port	BIOTQ	170	129	I/O
D_21_port	BIOTQ	167	130	I/O
D_20_port	BIOTQ	165	131	I/O
VCCB8	BVCCB	163	132	VDB
D_19_port	BIOTQ	160	133	I/O
D_18_port	BIOTQ	158	134	I/O
D_17_port	BIOTQ	156	135	I/O
D_16_port	BIOTQ	153	136	I/O

Name	Pad type	Pad number	Pin number	Pin type
VSSB9	BVSSB	151	137	VSB
D_15_port	BIOTQ	149	138	I/O
D_14_port	BIOTQ	146	139	I/O
D_13_port	BIOTQ	144	140	I/O
D_12_port	BIOTQ	142	141	I/O
VCCB9	BVCCBSNQ	139	142	VDB
VCCA9	BVCCA	137	143	VDA
VSSA9	BVSSA	135	144	VSA
VSSB10	BVSSBENQ	132	145	VSB
D_11_port	BIOTQ	130	146	I/O
D_10_port	BIOTQ	128	147	I/O
D_9_port	BIOTQ	126	148	I/O
D_8_port	BIOTQ	123	149	I/O
VCCB10	BVCCB	121	150	VDB
D_7_port	BIOTQ	119	151	I/O
D_6_port	BIOTQ	116	152	I/O
D_5_port	BIOTQ	114	153	I/O
D_4_port	BIOTQ	112	154	I/O
VSSB11	BVSSB	109	155	VSB
D_3_port	BIOTQ	107	156	I/O
D_2_port	BIOTQ	105	157	I/O
D_1_port	BIOTQ	102	158	I/O
D_0_port	BIOTQ	100	159	I/O
VCCB11	BVCCBSNQ	98	160	VDB

APPENDIX A: ABBREVIATIONS

ADC	Analog to Digital Converter
AGC	Automatic Gain Control
AGGA 2	Advanced GPS/GLONASS ASIC, second generation
AOCS	Attitude and Orbit Control System
ASIC	Application Specific Integrated Circuit
C/A code	Coarse/Acquisition code
CA channel	Channel that can only process C/A code

CaP channel	Channel that can process C/A -and P codes
CMOS	Complementary Metal Oxide Semiconductor
DSP	Digital Signal Processing
E	Early correlation
EGNOS	European Geostationary Navigation Overlay System
FIR	Finite Impulse Response
GLONASS	Global Orbiting Navigation Satellite System
GNSS	GNSS Global Navigation Satellite Systems
GPS	Global Positioning System
I	In phase
IF	Intermediate Frequency
I/O	Input/Output
JTAG	Joint Test Action Group
L	Late correlation
L1	Carrier frequency (GPS 1575.4 MHz, GLONASS 1602+ch·9/16 MHz)
L2	Carrier frequency (GPS 1227.6 MHz, GLONASS 1246+ch·7/16 MHz)
LFSR	Linear Feedback Shift Register
LSB	Least Significant Bit
MSB	Most Significant Bit
MSAS	Multi Transport Satellite Augmentation System
NCO	Numerically Controlled Oscillator
P	Punctual correlation
P code	Precision code
Q	Quadrature
RA	Regional Augmentation
RF	Radio Frequency
SV	Space Vehicle
TTL	Transistor Transistor Logic
UTC	Universal Time Coordinated
W code	Code used to encrypt the GPS P code to form the Y code
WAAS	Wide Area Augmentation System
Y code	Encrypted GPS P code

Appendix B: Reference Documents

- **RD1** ***NAVSTAR GPS Space Segment Navigation User Interface*****s, ICD-GPS-200 Rev. C / IRN-200C-002, September 1997, Arinc Research Corporation, USA**
- **RD2** ***Interface Control Document, Global Satellite Navigation System GLONASS*****,November 1995, Glavkosmos, Moscow, Russia**
- **RD3** ***Inmarsat-GNOS SD*****M, Issue 1, Annex 2, pp 1-35**
- **RD4** ***Description of Invention: Technique for Codeless Tracking of GPS Signal*****s, ref.JP/97-07-1517/PS, September 1997, European Space Agency**
- **RD5** ***GNSS Receiver Design for Attitude Determinatio*****n, Proceedings of Third ESA International Conference on Spacecraft Guidance, Navigation and Control Systems, ref. SP-381, pp. 275-285, February 1997, European Space Agency**
- **RD6** ***ADSP21020 32/40-Bit IEEE Floating-Point DSP Microprocesso*****r, Data Sheet Rev. C, August 1994, Analog Devices, USA**
- **RD7** ***TSC691E Integer Unit: User's Manua*****l, Rev. H, December 1996, TEMIC Semiconductors, France**
- **RD8** ***TSC692E Floating Point Unit: User's Manua*****l, Rev. H, December 1996, TEMIC Semiconductors, France**
- **RD9** ***TSC693E Memory Controller: User's Manua*****l, Rev. D, April 1997, TEMIC Semiconductors, France**
- **RD10** ***GP2010 GPS Receiver RF Front En*****d, Data Sheet ref. DS4056-3.4, October 1996, MITEL Semiconductor, United Kingdom**
- **RD11** ***MG2RT Radiation Tolerant 0.5-∝m CMOS Sea-of-Gates Data Shee*****t, Rev. B, February 1998, TEMIC Semiconductors, France**
- **RD12** ***MG2RT ASIC 3V LIBRARY Data Boo*****k, ref. ATD-TS-LR-R0216, rev. 1.2, June 1998, TEMIC Semiconductors, France.**

Ordering Information

Reference Number	Temperature Range	Package	Quality Samples
T7905EFW-E	+25°C	MQFPL160	Engineering Samples
T7905EFW	-55 to +125°C	MQFPL160	MIL
T7905EFW/883*	-55 to +125°C	MQFPL160	MIL 883 B
T7905EFWS/883*	-55 to +125°C	MQFPL160	MIL 883 S
T7905EFWSC	-55 to +125°C	MQFPL160	SCC C
T7905EFWSB	-55 to +125°C	MQFPL160	SCC B
T7905EFWMQ	-55 to +125°C	MQFPL160	QML Q
T7905EFWSV	-55 to +125°C	MQFPL160	QML V
T7905EDD-E	+25°C	Die	Engineering samples
T7905EDDMQ	-55 to +125°C	Die	QML Q
T7905EDDSV	-55 to +125°C	Die	QML V

Note: (*) contact factory

Atmel Wireless & Microcontrollers Sales Offices

France
3, Avenue du Centre
78054 St.-Quentin-en-Yvelines
Cedex
France
Tel: 33130 60 70 00
Fax: 33130 60 71 11

Germany
Erfurter Strasse 31
85386 Eching
Germany
Tel: 49893 19 70 0
Fax: 49893 19 46 21

Kruppstrasse 6
45128 Essen
Germany
Tel: 492 012 47 30 0
Fax: 492 012 47 30 47

Theresienstrasse 2
74072 Heilbronn
Germany
Tel: 4971 3167 36 36
Fax: 4971 3167 31 63

Italy
Via Grosio, 10/8
20151 Milano
Italy
Tel: 390238037-1
Fax: 390238037-234

Spain
Principe de Vergara, 112
28002 Madrid
Spain
Tel: 3491564 51 81
Fax: 3491562 75 14

Sweden
Kavallerivaegen 24, Rissne
17402 Sundbyberg
Sweden
Tel: 468587 48 800
Fax: 468587 48 850

United Kingdom
Easthampstead Road
Bracknell, Berkshire RG12 1LX
United Kingdom
Tel: 441344707 300
Fax: 441344427 371

USA
2325 Orchard Parkway
San Jose
California 95131
USA-California
Tel: 1408441 0311
Fax: 1408436 4200

1465 Route 31, 5th Floor
Annandale
New Jersey 08801
USA-New Jersey
Tel: 1908848 5208
Fax: 1908848 5232

Hong Kong
77 Mody Rd., Tsimshatsui East,
Rm.1219
East Kowloon
Hong Kong
Tel: 85223789 789
Fax: 85223755 733

Korea
Ste.605,Singsong Bldg. Young-
deungpo-ku
150-010 Seoul
Korea
Tel: 8227851136
Fax: 8227851137

Singapore
25 Tampines Street 92
Singapore 528877
Rep. of Singapore
Tel: 65260 8223
Fax: 65787 9819

Taiwan
Wen Hwa 2 Road, Lin Kou
Hsiang
244 Taipei Hsien 244
Taiwan, R.O.C.
Tel: 88622609 5581
Fax: 88622600 2735

Japan
1-24-8 Shinkawa, Chuo-Ku
104-0033 Tokyo
Japan
Tel: 8133523 3551
Fax: 8133523 7581

Web site
http://www.atmel-wm.com