UCCP Spectrum Analyser

Design Specification

Copyright © Imagination Technologies Ltd. All Rights Reserved.
This document is for internal use only. Neither the whole nor any part of the information contained in, nor the product described in, this document may be adapted or reproduced in any material form except with the written permission of Imagination Technologies Ltd. Distribution outside of Imagination Technologies Ltd is strictly forbidden.

Filename	:	UCCP Spectrum Analyser.Design Specification.docx
Version	:	1.0.40 Not Issued - Live Document
Issue Date	:	17 Sep 2013
Author	:	Imagination Technologies Ltd

Contents

1. Introduction 3

2. Signal Path and Control Architecture 3

2.1. Measurement at DC 5
2.2. Gain Control 7

3. Software Structure 7

3.1. MeOS Tasks 8
3.2. Control State Machine 8

4. References 10

List of Figures

Figure 1: MCPOS Queues 4
Figure 2: DC Compensation 6
Figure 3: Control State Machine 9

IF{{Internal}}

Document History

Issue	Date	Changes/Comments

END{{Internal}}

1. Introduction

This document describes the proposed software design of a spectrum analysis system to run on the UCC platform (UCCP 330 onwards). The main purpose of this system is to support fast scanning of an area of spectrum to determine what TV transmissions might be present. It may be used alongside any existing or future TV demodulator system.

Products incorporating our TV demodulators will normally include a "blind scan" operation which is initiated when the product is first connected or if the product is moved to a new area. This spectrum analysis system can be used as the first stage of such a scan, supplying carrier frequencies (and if required, bandwidths) for any candidate transmissions. The TV demodulator can then be directed at these specific candidates to determine a) whether they are valid signals and b) what the signal parameters are. This is generally much quicker than trying to use demodulator application itself to search across a wide range of carrier frequencies (and symbol rates, in the case of Satellite).

Implementation of this system has become necessary because of a requirement to scan satellite bands in a relatively short time. Using the demodulator (PHY) to perform the scan in this case is particularly slow, firstly because the scan space is 2-dimensional (carrier frequency + baud rate); and secondly because the PHY is very slow to acquire (and thus to declare if a signal is present or not) when configured for low symbol rate. These issues are discussed in more detail in [1].

Note that blind scan operations are normally multi-standard: i.e. the user will want to detect all standards which are supported by a particular system / tuner. This will involve using a number of different demodulators to examine the candidate frequencies identified by the spectrum analyser.

In addition to supporting scan operations, this system may be used for a number of other purposes, for example:

- Detecting signalling, such as "UniCable" signalling between an LNB and a Satellite receiver
- Measuring tuner performance.

The system may also be of particular use in terrestrial applications, allowing detection of high-power adjacent channels (this situation otherwise has to be inferred by the PHY core).

2. Signal Path and Control Architecture

The signal path for this system is very simple, consisting only of the SCP feeding captured data from the ADCs into GRAM.

The captured data from the SCP is processed by MCP code, which windows the data, runs an FFT on each captured data block, calculates spectral power, and averages the result into output buffers (also in GRAM). This process is repeated at different tuning positions, obtaining a series of ***spectral fragments*** which can be pieced together to form the overall spectrum.

The SCP will be controlled by the standard SCP DCP driver, with its job queues being serviced by the MCP code. When not supplied with any jobs, the SCP will continually run default discard jobs. When the MCP wishes to receive new captured data, it queues a job into the SCP driver's input queues. The MCP code waits on the SCP driver's output queues in order to determine that the new captured data is available.

The SCP output interfaces to two GRAM buffers, of length set by the largest FFT size supported (8192 samples). This double-buffering allows the SCP to be capturing new data while previous data is being processed by the MCP. Note however that there is no requirement for the MCP processing to keep up with the speed of data input through the SCP. If the MCP takes longer than the capture time, then posting of a subsequent job will be delayed and the SCP driver will run a discard job. This does not pose a problem since there is no requirement for consecutive FFTs to be on contiguous data blocks.

The MCP code runs under MCPOS. Connections to the MCPOS OR gate are arranged as shown in Figure 1:

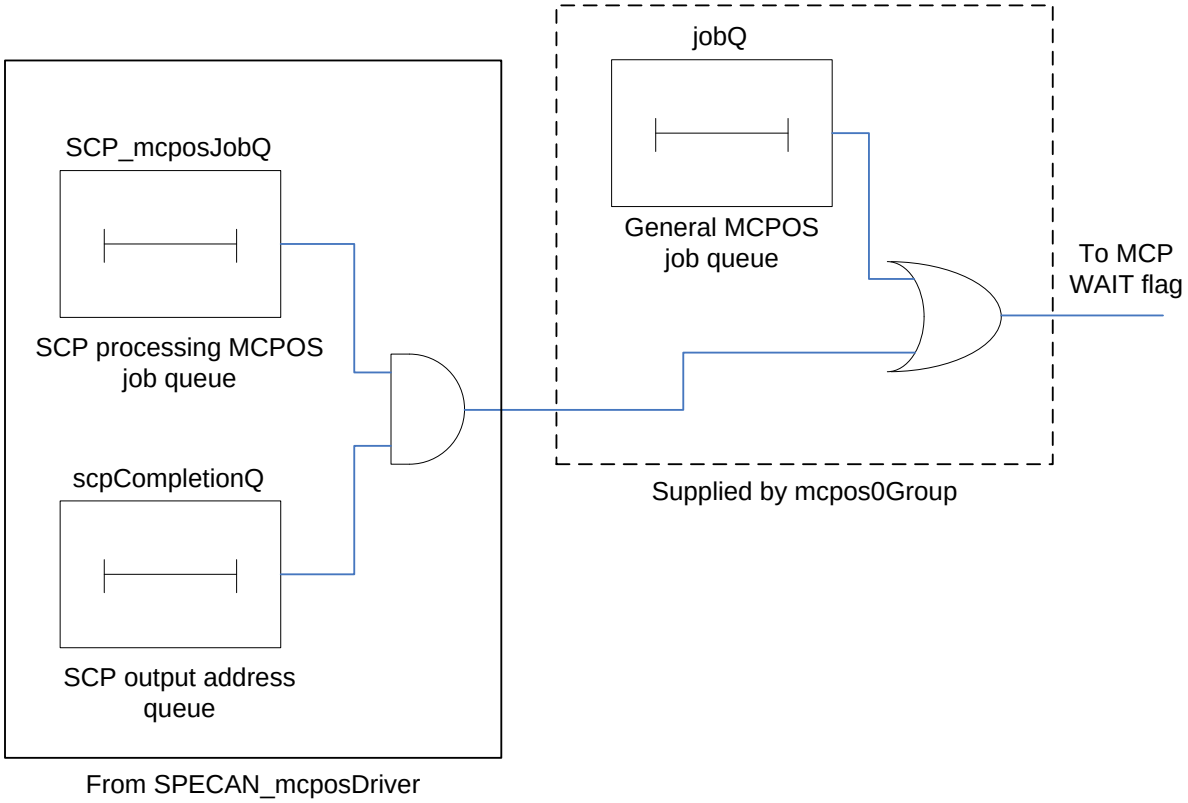

Figure 1: MCPOS Queues

Each time an SCP job is queued, an associated MCPOS job is also queued into SCP_mcposQ. Thus when the SCP job completes, it posts into its output address queue (scpCompletionQ), and the MCPOS OR will be activated. vector=true is specified when instantiating the MCPOS queue, which ensures that we read the job from the MCPOS queue rather than the SCP completion queue when MCPOS is activated.

The other MCPOS job queue (jobQ, which is instantiated by MCPOS as standard) could be used for queueing MCPOS jobs which are not associated with SCP completion, if necessary.

Capturing of each spectral fragment is kicked off by Meta which queues two SCP jobs into the SCP driver's input queues, and also queues two associated MCPOS jobs. When these jobs complete they kick off MCP code to process the SCP output data.

The MCP code runs a user-selectable window across the data, performs an FFT of a pre-set length on the windowed data, and calculates a spectral power vector by multiplying each bin value by its conjugate.

The MCP code at the end of an MCPOS job is responsible for removing jobs from the output queues and queueing new jobs into the input queues, in order to maintain a flow of buffers as required by the averaging process. The averaging is performed using two nested loops, with the inner loop running for N iterations and the outer loop running for M iterations. Within the inner loop, the power data from each FFT is scaled by 1/N and summed into an intermediate buffer. Within the outer loop, the inner loop result is scaled by 1/M and summed into the result buffer. This accomplishes an averaging over N * M buffers, while avoiding any quantisation issues that would result from scaling the data down by N * M in one go.

Once the averaging is complete, Meta will be interrupted by the MCP code. The MCP code will cease queuing new SCP jobs, so MCP processing will then halt until the next fragment processing is started by Meta.

Note that the frequency resolution of the spectral analysis is controllable via the API. Meta code will translate this into a suitable FFT size, effective sample rate and SCP decimation factor (given knowledge of the IF clock frequency). The amount of averaging (noise reduction) can be controlled using the parameters N and M above.

Meta code will be responsible for running a scan across a frequency band. This will involve a repeated process of tuning to a new frequency, capturing a spectral fragment, then building up a spectral vector for the whole band by piecing these fragments together. Note that only the lowest 80% of the frequency bins from each capture will be used. This is because the anti-alias filtering (both analog, in the tuner filter, and digital in the case of the SCP's resampling / decimation filters) will have a transition band which will be visible in the highest frequency bins (reaching a high level of attenuation at Nyquist). This means that the frequency ranges covered by each capture will need to overlap.

Via the API, the core is informed of a tuning frequency step to use as well as start frequency and scan range. The API also sets up tuner bandwidth and frequency resolution for the scan. For satellite scanning, the tuning frequency step can be set to "auto" in which case it is worked out by the spectrum analyser core as detailed above. For terrestrial scanning, the frequency step is likely to match the channel spacing; so the scan looks specifically at each channel, using a tuner bandwidth appropriate for the channel. The SCP's resample factor and decimation factors will be controlled by Meta to translate the ADC clock rate into the required effective sample rate.

The core will also be made aware of the tuning grid, so that it has accurate knowledge of the actual tuned frequency, allowing it to accurately piece the different spectral fragments together.

2.1. Measurement at DC

With a zero-IF tuner, DC falls in the middle of the spectral fragment being analysed. This is a potential problem because the DC measurement will be affected by any residual DC introduced by the analog circuits prior to the ADCs.

One possible solution to this is to halve the size of a spectral segment, only using just under half of the bins from each FFT result; thus we can avoid using the DC bin altogether. However this will increase the length of time for a scan.

A preferable approach is to use two different tuning frequencies for each spectral segment, spaced a small distance apart, each side of the nominal centre of the segment. Half of the FFT accumulations are made at one tuner frequency, half at the other. In each case, the LO in the SCP is adjusted to

compensate for the tuner frequency offset, so that the spectrum seen at the SCP output remains the same, except that different FFT bins are affected by the DC at the ADCs in each case. This is illustrated in Figure 2.

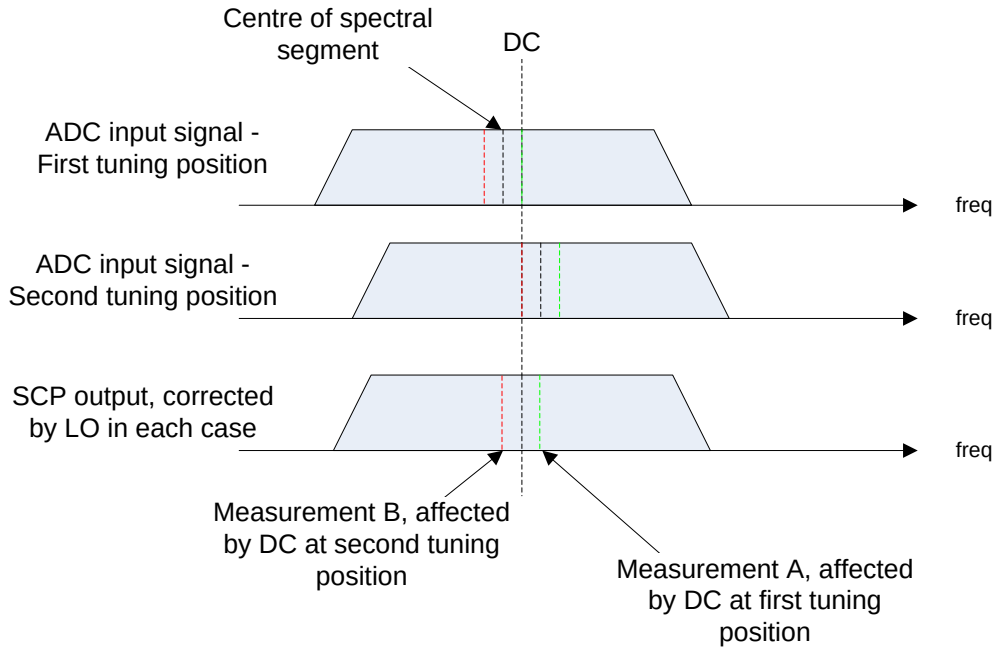

Figure 2: DC Compensation

All the frequency bins apart from bins A and B can be averaged in the usual way. The treatment of bins A and B is as follows:

- We have measurements as follows: A1 (bin A for first tuning position, affected by DC); A2 (bin A for second tuning position, unaffected); B1 (bin B for first tuning position, unaffected); B2 (bin B for second tuning position, affected by DC).
- The offset caused by DC in the two sets of measurements should be the same; so we can estimate it as DCest = ((A1 – A2) + (B2 – B1))/2
- We can then subtract this estimated offset from the corrupted DC measurements A1 and B2 before completing the averaging for these two bins. i.e. A = (A1 – DCest + A2)/2 and B = (B2 – DCest + B1)/2.

This correction process is managed by Meta without the knowledge of the MCP code; i.e. it requests two averaging results from the MCP sub-system, using different tunings, and makes the final correction for itself (including doing the averaging of the two results for all the other bins).

This feature can be turned off for the case where we have a real IF input.

Note that a DC control loop will not be run. The only advantage of such a control loop might be to give better headroom inside the FFT. As the DC component has already been digitised by the ADC before we run the control loop, we cannot do anything to improve the SNR at the SCP input by removing DC, and the next noise “bottleneck” in the system is the FFT. A strong DC component will resolve coherently through all the stages of the FFT, so the scaling will have to be pessimistic to account for this, reducing the dynamic range at the FFT output. However, the requirement for dynamic range is not particularly tight so it should be OK to just accept the DC offset that we get from the tuner.

2.2. Gain Control

After tuning and prior to making each analysis of a spectral segment, the Meta code will run a standard front-end AGC on the signal in order to establish a suitable analog gain which optimises the use of the number range available at the SCP input. (Note that, although the MCP will not be queueing any jobs at this point, the SCP will be running discard jobs so data will be running through it and the clip counters will be operational). Prior to running the spectral analysis of the segment, the analog gain setting is frozen so no gain changes will occur during the analysis.

This will result in different gain settings for each spectral segment (which is desirable as it allows us to support the desired dynamic range in spectral power). Meta is able to compensate for these different gain settings because the spectral fragments overlap, so that some frequencies appear in two neighbouring segments. Comparing the measurements for these frequencies between the two segments allows the actual gain change, as seen in the digital domain, to be quantified; such that Meta is then able to post-process the results, scaling them so they all have the same effective final gain.

In the case of a terrestrial tuner, there is an overlap between frequency bins from adjacent spectral fragments even if each tuning represents the centre of a channel. This is because the tuner's channel filter only starts rolling off at the edge of a channel; so the bins falling into the adjacent channel still have useful content (although it has been attenuated by the channel filter). If the channel filter's characteristics are known and communicated to the system via the API, then this attenuation can be allowed for.

The digital gain through the SCP will be left fixed.

Note that the spectral power values supplied by this application are only useful in a relative sense; there is no attempt to translate to an absolute received power. This is particularly true in the case of terrestrial scanning where the tuner may have applied an RF AGC whose operation is not visible to our system. This does not really matter, since the purpose of the system is to characterise what the spectrum will look like to a PHY core which is viewing the same tuner output.

3. Software Structure

To minimise software design and maintenance, the system will use the TV appShell which brings in the common API registers and state machine [2]. This state machine doesn't really fit the model of how this core operates. However, we can construct a sensible-looking user interface by only using a few of the states and by producing alternative documentation that re-names the states and omits to mention unused features. Thus our state machine may only use the states DORMANT, INITIALISED DETECTING and ACQUIRING, and the documentation can rename the ACQUIRING state to COMPLETED.

The user model is thus as follows: the core is activated or de-activated in the usual way and uses the standard virtual register system. After activating, the user sets up scan parameters via the API then issues a DETECT command. This moves the state machine into the DETECTING state. When the scan is complete the state machine moves into the COMPLETED state to indicate that the scan is complete. The user issues a STOP command once the scan data has been processed, and can then either initiate another scan or de-activate the core.

The output frequency data is presented as a pointer to a vector of power values in memory, with the index within the vector translating to frequency according to the base frequency and frequency resolution values originally supplied by the user application. The pointer is supplied to the user in the form of a memory type indication (GRAM or ext. RAM) and an offset within the memory. This is

preferable to an absolute address because the user application may be using a different address map from Meta.

The software also has the capability to report indices of the 8 highest-power frequencies present, along with their associated power values, directly in API registers.. This can be used in place of obtaining a complete spectrum, for example in order to detect unicable signalling within a narrow part of the spectrum. For this analysis, bins immediately adjacent to a local maximum are ignored (in order that each of the 8 results represents an independent peak).

The system will run under MeOS as is expected for all cores following the appShell model.

3.1. MeOS Tasks

Excluding any application tasks, there will be 3 tasks running within the spectrum analyser core:

- "TV Core Main Task" forms part of the appShell system, this runs the common API state machine
- "SpecAn Control Task" is our main controlling task; this responds to requests for action, placed into a mailbox. These requests may come from TV core main task, or from an ISR responding to events on the MCP. This task queues MCPOS jobs and controls the scan process. It raises events for the TV Core Main Task to respond to, resulting in API state transitions.
- "Front End Task" runs the analog AGC, when active. It responds to interrupts from the SCP's AGC counters. It is only active when enabled by the SpecAn Control Task.

3.2. Control State Machine

The control task implements a state machine as per Figure 3. The system sits in the IDLE state while not performing a scan. When a scan is requested, it moves forward into the START state. Here fixed parameters are set up, according to the requested scan parameters, such as resample factor,decimation factor, FFT size etc.

On completing these operations the state machine immediately moves forward into the TUNE state.

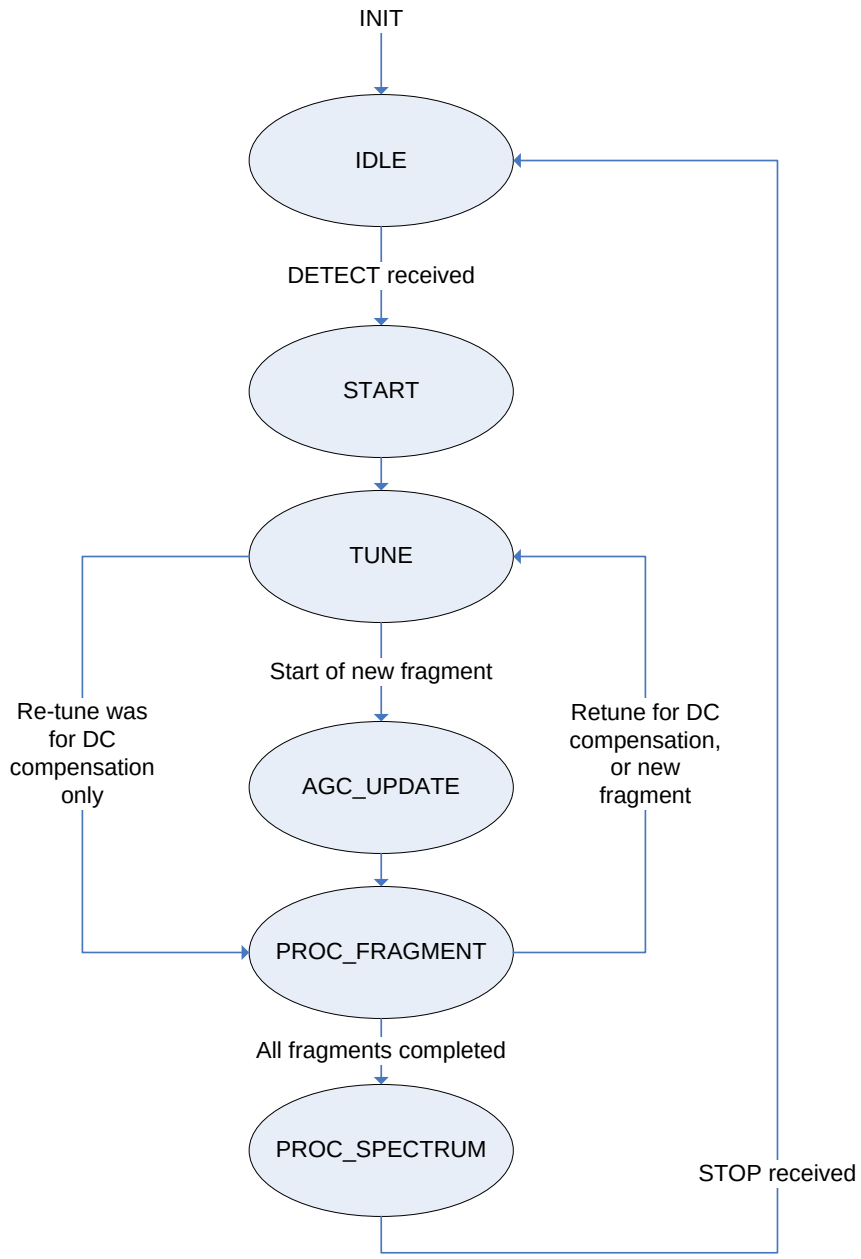

Figure 3: Control State Machine

On entering the TUNE state, a re-tune is performed to move to the correct point in the spectrum. On receiving the TUNED event, the state machine then moves forward into AGC_UPDATE, where the front-end AGC is started. The AGC runs for a preset number of interrupts, sufficient for the AGC control loop to stabilise. The front end task then messages the control task, causing a move forward into PROC_FRAGMENT. At this point the AGC is stopped, and the MCP-based fragment processing is kicked off as described in section 2.

If DC compensation, as described in section 2.1, is enabled then the system jumps back to the TUNE state at this point, re-tuning by a small amount and adjusting the LO accordingly. It then moves back to PROC_FRAGMENT in order to complete the rest of the averaging associated with the fragment.

As each fragment processing is completed the state machine then moves back to TUNE, re-tuning to the next position within the spectrum, and the process repeats for as many times as there are fragments in the spectrum. Once all the tuning positions have been covered, it moves to

PROC_SPECTRUM. Here any post-processing is performed on the spectral data, and a COMPLETE event is raised to the API, alerting the user that results are available.

If a STOP command is received from the API at any point in the processing then the state machine returns to IDLE.

4. References

[1] Satellite Scanning.White Paper.docx (IMG white paper 10/10/2012)

[2] UCCP Common TV API.Virtual Register Interface.doc