Multibunch feedback application software

A. F. D. Morgan

March 3, 2022

Contents

Chapter 1

Overview

The hardware for multi bunch feedback has many different possible settings. Broadly speaking feedback system has a frontend which takes in the signals from the pickups, and does the initial analogue signal conditioning, connected to digital systems for each of the three planes (x,y,s). These digital systems calculate the feedback required and send signals to analogue drive electronics in order to excite the beam in the given plane.

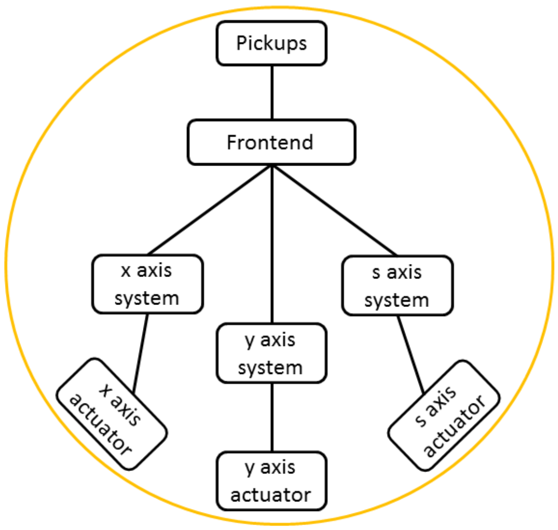


Figure 1.1: System overview

In order to make such a system more generally useable, this suite of task specific applications has been developed. All these codes save the relevant data and graphs to a file structure based on the date, and given unique names based on the date and time the application was run. The root path for this structure is defined in `mbf_system_config.m`, as is the harmonic number of the machine under study. In general all machine/deployment parameters are set in this file. The rest of the codebase it written to adapt to what is set here.

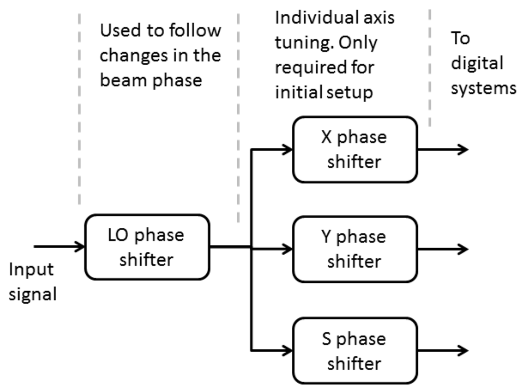
Additional to the data from the multi bunch hardware, more general machine parameters are also captured for growdamp, modescan and bunch motion codes, in order to enable machine studies (e.g current dependance). This additional data is set in `machine_environment.m`.

Chapter 2

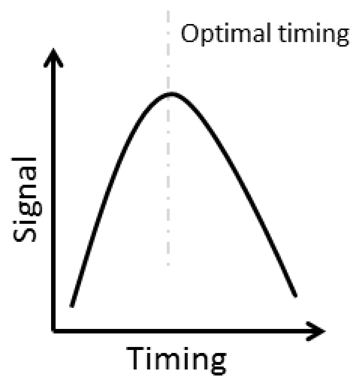
Frontend phase scans

In order to work optimally the feedback systems need to be matched in timing/phase with the beam phase (not the RF phase). Conceptually this comprises a master phase shifter (LO) which moves all three systems and allows the complete system to be adjusted to changes in beam phase, combined with three subsidiary phase shifters, one for each axis to allow the axes to be individually tuned in order to account for differences in the phase delay of the different chaannels.

The practical implementation will depend on the details of your frontend hardware. For example you may have automatic beam phase following which renders the LO phase scans largely redundent. Additionally, the timing of the clocks in the digital part of the system need to be


Used to follow changes in the beam phase
Individual axis tuning. Only required for initial setup
To digital systems
Input signal
LO phase shifter
X phase shifter
Y phase shifter
S phase shifter

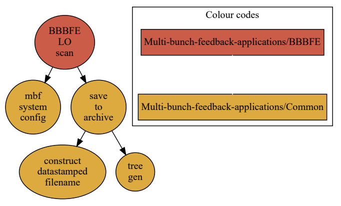
(a) phase shifters



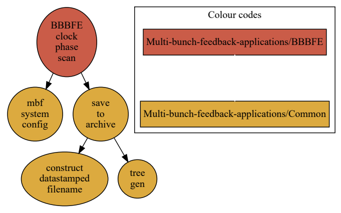
(b) Clock timing

Figure 2.1: Conceptual diagrams

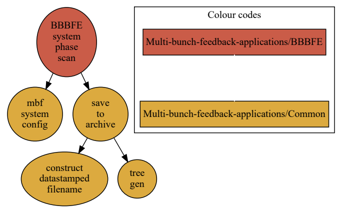
tuned so that the sampling point coincides with the maximum signal from each bunch. This both increases the sampled signal, but also minimises the leakage to adjacent bunches.


BBBFE LO scan
mbf system config
save to archive
construct datastamped filename
tree gen
Colour codes
Multi-bunch-feedback-applications/BBBFE
Multi-bunch-feedback-applications/Common

(a) LO scan


BBBFE clock phase scan
mbf system config
save to archive
construct datastamped filename
tree gen
Colour codes
Multi-bunch-feedback-applications/BBBFE
Multi-bunch-feedback-applications/Common

(b) Clock phase scan


BBBFE system phase scan
mbf system config
save to archive
construct datastamped filename
tree gen
Colour codes
Multi-bunch-feedback-applications/BBBFE
Multi-bunch-feedback-applications/Common

(c) System phase scan

Figure 2.2: Code trees for bunch by bunch frontend scans

Typical output:

Often you can just use `BBBFE_LO_scan` to tune all four phase shifters, as one is aiming to have the peaks of the three curves at the same phase angle.

Note: The Libera bunch by bunch frontend seems to get into problems when scanning the LO phase shifter. It is probably best not to use `BBBFE_LO_scan` if you have that frontend.

2.1 Examples

Scan the local oscillator 45 degrees either side of 90 degrees. This is the first check to see if the three systems are aligned to the beam phase.

```
BBBFE_LO_scan(90, 45)
```

The next two examples are done on an individual axis at a time. Sort out the mapping of clocks to axes.

```
Setup: (manual for the time being...)
press tune only
fill some charge in bunch 1 (0.2nC)
set bunch mode single bunch
set single bunch 1
press setup tune
set up individual tune detectors to run on the selected bunch and one bunch either side.
set sweep gain to -18
set detector fixed gain
```

```
BBBFE_clock_phase_scan(1)
```

```
BBBFE_system_phase_scan('X')
```

Chapter 3

Mode scans

A mode scan is a sequential excitation of the harmonics of the tune frequency. This is mainly a system check in order to identify any performance limitations or degradation.

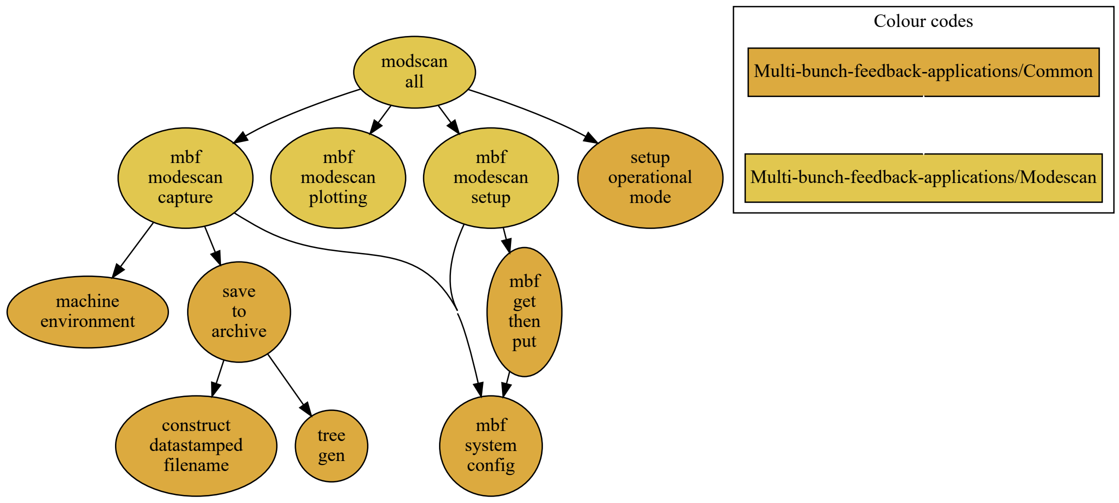


Figure 3.1: Modescan code tree

3.1 Examples

```
mbf_modescan_all('x')
```

Chapter 4

Spectra

This code captures the requested number of turns from the requested axis and returns the FFT of the centroid motion of each bunch.

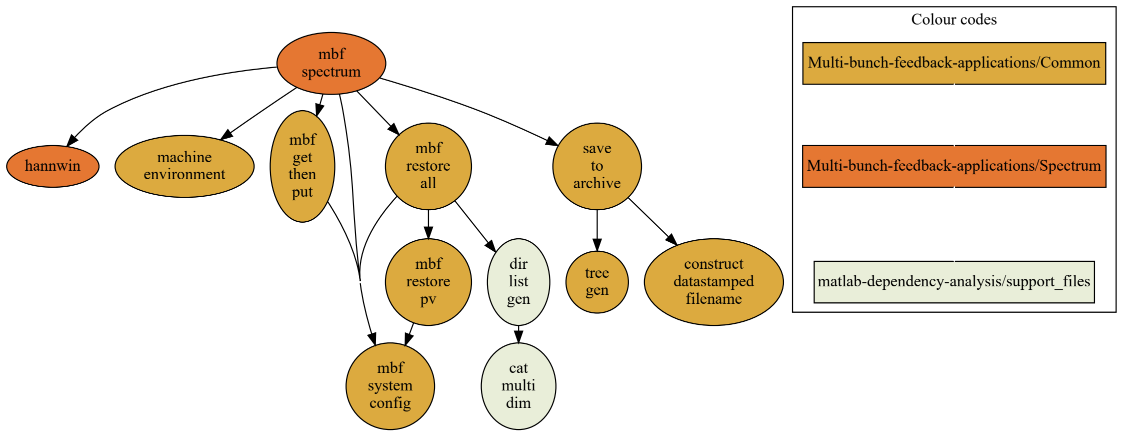


Figure 4.1: Spectra code tree

Typical output:

4.1 Examples

```
[f_magnitude,f_phase] = mbf_spectrum('x', 100)
```

Chapter 5

Growdamp

The measurement comprises a series of excitations at set harmonics of the tune frequency followed by capture of the evolution of bunch centroid positions, both with and without feedback active.

5.1 Capturing data

The parameters for the length of excitation and capture are set in `mbf_growdamp_config.m`. The `mbf_growdamp_setup` code sets up the hardware for the correct type of measurement. `mbf_growdamp_capture` triggers the measurement, captures the data and stores it in the file structure defined in `mbf_system_config`. Finally `mbf_restore_all` returns the settings for the MBF system to their original states.

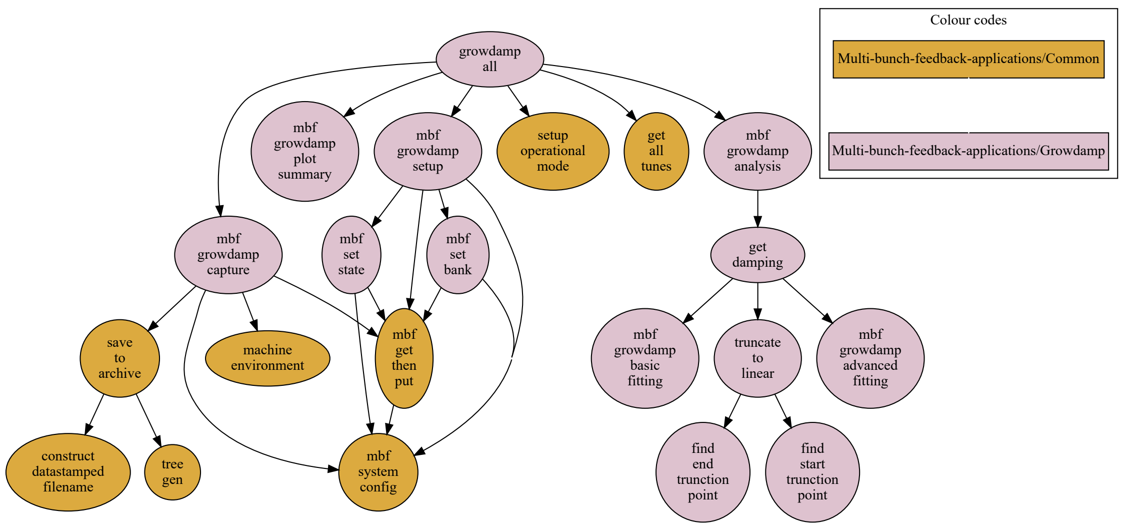


Figure 5.1: grow damp code tree

Chapter 6

Transverse damping

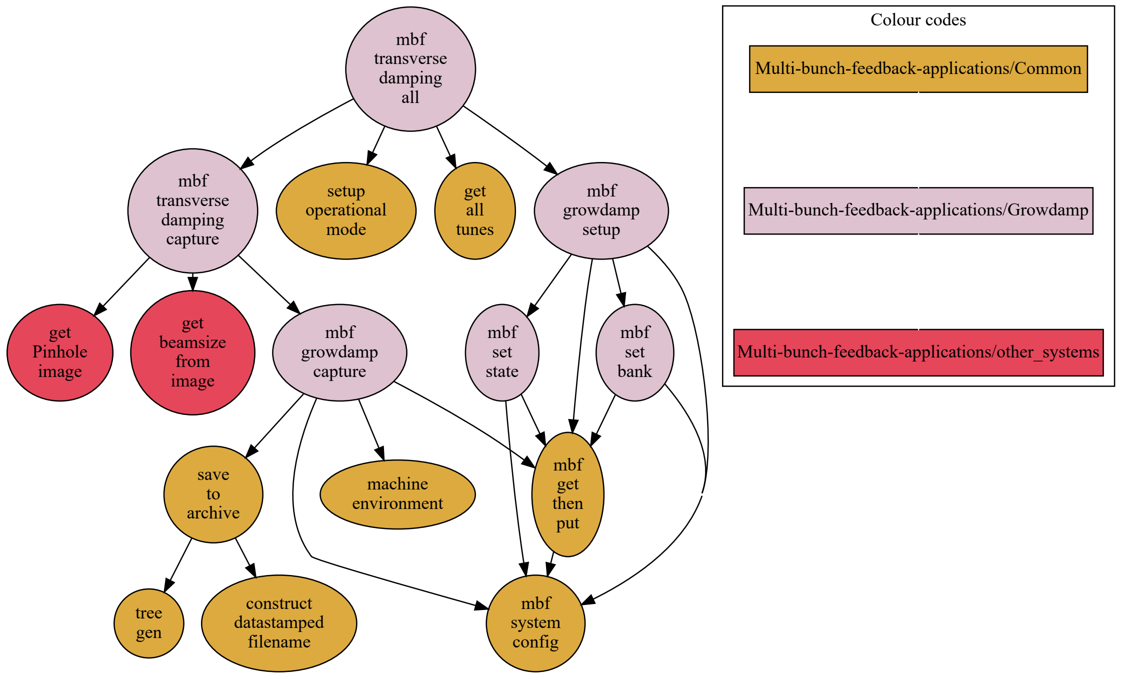


Figure 6.1: Transverse damping code tree

6.1 Examples

Chapter 7

Emittance control

This uses one of the oscillators to excite the selected bunches at one of the sideband frequencies. This has the effect of blowing up the beam.

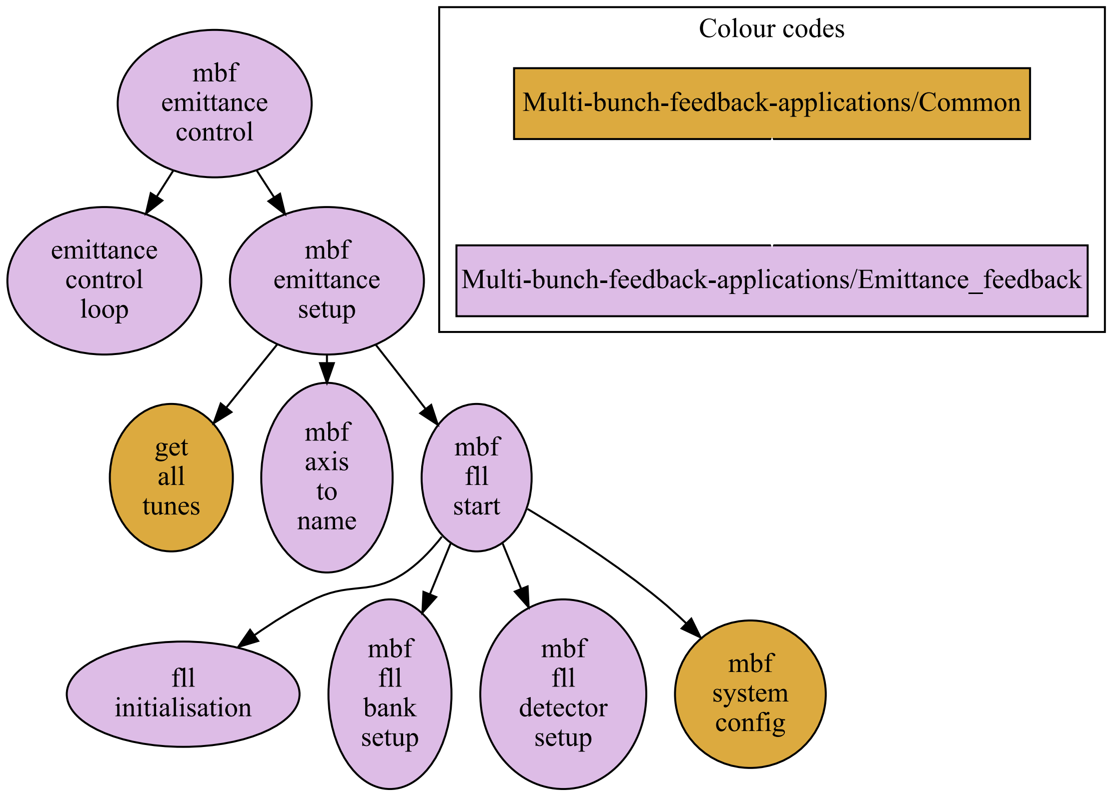


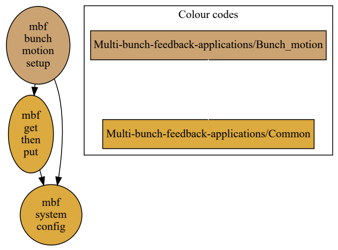
Figure 7.1: Emittance control code tree

7.1 Examples

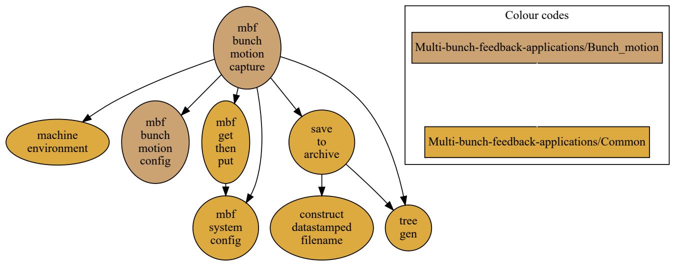
Chapter 8

Bunch motion capture

The three systems are setup to be ready to capture on the next external trigger. Once that trigger arrives, the three systems simultaneously capture bunch centroid motion. This is useful for capturing injection transients. `mbf_bunch_motion_setup` sets up all three systems to use the appropriate triggers. `mbf_bunch_motion_capture` arms the systems, and once triggered by an external trigger, will capture and store the data. As with the growdamp application, `mbf_restore_all` will bring the systems back to their original state.


mbf bunch motion setup
mbf get then put
mbf system config
Colour codes
Multi-bunch-feedback-applications/Bunch_motion
Multi-bunch-feedback-applications/Common

(a) Code tree for bunch motion setup



(b) Code tree for bunch motion capture

Figure 8.1: Code trees bunch motion

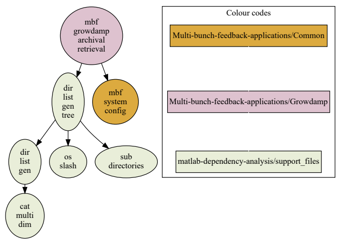
8.1 Examples

```
mbf_bunch_motion_setup
bunch_motion = mbf_bunch_motion_capture
mbf_restore_all
mbf_bunch_motion_plotting(bunch_motion)  <--- Experimental!
```

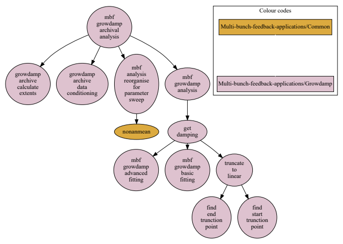
Chapter 9

Analysing saved data

Although one can analyse the data immediately using `mbf_growdamp_analysis`, it is often more useful to analyse a set of previously captured data. `mbf_growdamp_archival_retrival` will return all the data sets within a user defined time period. These data sets can then be passed to `mbf_growdamp_archival_analysis` where the data can either just be plotted for visual comparison, be averaged, or to be plotted against one of the machine parameters also stored in the data sets.


mbf growdamp archival retrieval
dir list gen tree
mbf system config
dir list gen
os slash
sub directories
cat multi dim
Colour codes
Multi-bunch-feedback-applications/Common
Multi-bunch-feedback-applications/Growdamp
matlab-dependency-analysis/support_files

(a) For retriving data sets from the data store


mbf growdamp archival analysis
growdamp archive calculate extents
growdamp archive data conditioning
mbf analysis reorganise for parameter sweep
mbf growdamp analysis
nonanmean
get damping
mbf growdamp advanced fitting
mbf growdamp basic fitting
truncate to linear
find end trunction point
find start trunction point
Colour codes
Multi-bunch-feedback-applications/Common
Multi-bunch-feedback-applications/Growdamp

(b) For analysing data sets (display, averaging, parameter scans)

Figure 9.1: Code trees grow damp archival analysis

Typical output:

9.1 Examples

Capture (and plot) x axis data.

```
tune = 0.17 <-- This could also be retrieved from a lcaGet.
mbf_growdamp_setup('x', tune)
growdamp = mbf_growdamp_capture('x', tune);
mbf_restore_all
[poly_data, frequency_shifts] = mbf_growdamp_analysis(growdamp);
mbf_growdamp_plot_summary(poly_data, frequency_shifts)
```

Capture ten sets of x axis data.

```
tune = 0.17 <-- This could also be retrieved from a lcaGet.
mbf_growdamp_setup('x', tune)
for ha = 1:10
    mbf_growdamp_capture('x', tune);
```

```
end
mbf_restore_all
```

Find and plot a single data set.

```
requested_data = mbf_growdamp_archival_retrieval('y', [requested_timestamp, requested_timestamp]);
mbf_growdamp_archival_analysis(requested_data, 'collate')
```

Find and plot a series of data sets for the y axis.

```
requested_data = mbf_growdamp_archival_retrieval('y', [now-5, now-1]);
mbf_growdamp_archival_analysis(requested_data, 'collate')
```

Gather all the y axis data sets between two times and plot the result of averaging them.

```
requested_data = mbf_growdamp_archival_retrieval('y', [now-5, now-1]);
mbf_growdamp_archival_analysis(requested_data, 'average')
```

Gather all the y axis data sets between two times and plot behaviour with beam current, averaging any sets which are closer than 20mA.

```
requested_data = mbf_growdamp_archival_retrieval('y', [now-5, now-1]);
mbf_growdamp_archival_analysis(requested_data, 'parameter_sweep', 'current', 20)
```