Control in Execution: Pulse Sequences to Waveforms & Real-Time Controllers

Maxim Zaitsev[1]
[1]*University Medical Centre Freiburg, Freiburg, Germany*

TARGET AUDIENCE: Researchers seeking a deeper understanding how MR pulse programs are transformed into electrical signals, as well as those willing to extend or compliment their hardware platforms with additional components and functionality.

INTRODUCTION: The notion of MRI console as a black box responsible for the interface between the sequence programmer or MRI device operator and the (historically predominantly analogue) electronics needed to actually perform a particular MRI experiment is quite common. Here we intend to look behind the scenes to discover how the gradient and RF pulse directives requested by the pulse sequence programmer are transformed into electric waveforms or how data acquisition commands are carried out, etc.

It is rarely the case that MR physicists build their own console electronics from scratch. However, the knowledge of the mechanisms and principles behind the system control may still be valuable in two respects. First of all, a detailed knowledge on how the system functions would allow one to understand and compensate for system imperfections. Secondly, understanding of the system architecture and interactions becomes essential if the existing control electronics needs to be extended to implement additional functions (e.g. parallel transmit, dynamic shimming, etc.)

In this presentation we build up a requirement specification of a complete MRI control subsystem starting from a basic FID experiment and increasing the complexity step-by-step. The primary goal of this contribution is to develop an understanding of the basic functionality of the particular components involved, but more importantly the interdependencies between them.

WE BEGIN with a minimalistic set-up to perform a pulse-acquire free induction decay (FID) MR experiment and find a minimal set of necessary components to excite the spin system and record an FID. Additional requirements arising from the need of using signal averaging, shaped RF pulses and multiple RF pulses within a single experiment are discussed in detail. This simple set-up is already very relevant as it allows one to implement several options of the current research areas, e.g. a concurrent field monitoring. Challenges and solutions associated with the extension to multiple transmit and receive channels are discussed.

THEREAFTER we complement our minimalistic console with a gradient controller and discuss the associated specifications. Design decisions for implementing eddy current compensation (ECC) and gradient delay calibration are discussed. A particular challenge is the compensation of the B0 eddy current, which often is the most significant component. At this juncture two principally different solutions are possible: by implementing a B0 compensation coil or by dynamically adjusting the carrier frequency.

The console described so far is sufficient for performing the majority of the standard MRI experiments. It may also be extended to include dynamic shimming, non-linear gradients, broad-band transmit option, etc. The gradient controller may also support various transformation and mixing modes to implement slice and FOV rotation, slice-specific shimming or eddy-current correction, etc.

ADDITIONAL INPUTS may be required to implement further features, primarily associated with in vivo imaging, such as triggering, gating, real-time feedback or safety monitoring. Different approaches of insuring gradient and RF safety of the subject are discussed.

We review briefly AVAILABLE DEVELOPER COSOLE solutions, both open-source and commercial ones, with the emphasis on the former ones and the interesting technical solutions involved.

A significant practical challenge in adopting new or alternative console solutions is a necessity of developing or learning a new SEQUENCE PROGRAMMING ENVIRONMENT and re-implementing the existing sequences for the specific console. This arises due to the fact that MR pulse sequences are commonly implemented on a very low level, which makes the instructions used very system-specific. Furthermore, to achieve the best possible performance in the given settings, console- and hardware-dependent limitations need to be taken into account and specialized optimisations have to be performed. Such strong ties to the console solutions have developed themselves historically, also in part because of the commercial interest behind preserving a certain amount of isolation between the platforms. We discuss several typical limitations of selected well-known platforms.

We ADVOCATE strongly for a development of a system-independent sequence programming environment, which output will be transformed to console-specific instructions by specialized programs and libraries. Similar hardware abstraction layers (HALs) are known from the common computer / operating system infrastructures, where the details of the low-level hardware implementation are hidden from the user-space programs. Development of the console-independent pulse programming language and environment will facilitate greatly exchange between the various research groups and transition between the consoles.